Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Kristian Žarn

Gradnja 3D modela s premikajočo se monokularno kamero

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: doc. dr. Danijel Skočaj

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Tipično se za gradnjo 3-dimenzionalnih modelov predmetov uporabljajo globinski senzorji ali stereo sistemi kamer. 3D model pa lahko zgradimo tudi z eno samo kamero s postopkom imenovanim struktura iz gibanja. V diplomski nalogi implementirajte ta postopek, ki omogoča redko rekonstrukcijo predmeta ter določitev zunanjih parametrov kamere. Le-te uporabite za gosto rekonstrukcijo predmeta z uporabo stereo tehnike. Izdelani postopek tudi ustrezno ovrednotite.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Kristian Žarn, z vpisno številko **63110352**, sem avtor diplomskega dela z naslovom:

Gradnja 3D modela s premikajočo se monokularno kamero

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom doc. dr. Danijela Skočaja,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 14. septembra 2015 Podpis avtorja:

Zahvaljujem se svojemu mentorju doc. dr. Danijelu Skočaju za pomoč in staršem za podporo.

Kazalo

Seznam uporabljenih kratic

kratica	angleško	slovensko
IR	infrared	infrardeč
RGB	red green blue color model	rdeč zelen moder barvni model
DLT	direct linear transform	algoritem neposredne linearne transformacije
SVD	singular value decomposition	razcep na singularne vrednosti
SIFT	scale invariant feature transform	transformacija v skalabilno invariantne značilnice
SURF	speeded up robust features	pohitrene robustne značilnice
BRISK	binary robust invariant scalable keypoints	binarne robustne invariantne skalabilne značilnice
FREAK	fast retina keypoint	hitre retina značilnice
RANSAC	random sample consensus	konsenz naključnih vzorcev
SAD	sum of absolute difference	vsota absolutnih razlik
ICP	iterative closest point	iterativna metoda najbližje točke

Povzetek

Za gradnjo 3D modelov so tipično uporabljeni globinski senzorji ali stereo sistemi kamer. Takšne rešitve so manj dostopne, zato je zanimiv izziv na področju računalniškega vida gradnja 3D modelov brez dodatne opreme. V diplomskem delu obravnavamo rekonstrukcijo predmetov na podlagi fotografij. Najprej pregledamo teoretično ozadje in uporabljene algoritme, nato pa predstavimo naš sistem za rekonstrukcijo. S postopkom imenovanim struktura iz gibanja, pridobimo redko rekonstrukcijo predmeta in zunanje parametre kamer. Slednje uporabimo pri gradnji gostega 3D modela. Evalvacija zunanjih parametrov pokaže, da pri dodajanju kamer v sistem pride do akumulacije napake. Povprečna napaka lege kljub temu znaša le nekaj milimetrov. Na koncu ovrednotimo še natančnost pridobljenih modelov tako, da jih primerjamo z referenčnimi. Rezultati pokažejo, da je povprečna napaka rekonstrukcij majhna, zato lahko povzamemo, da implementirani sistem deluje dovolj dobro.

Ključne besede: struktura iz gibanja, kalibracija kamere, redka rekonstrukcija, gosta rekonstrukcija, 3D model.

Abstract

Reconstruction of a 3D model is typically obtained using depth sensors or a stereo camera system. Such solutions are less accessible, therefore it is an interesting challenge in the field of computer vision to build a 3D model without additional equipment besides a camera. Our work discusses reconstruction based on images. First we examine the theoretical background and algorithms, then we proceed with description of our reconstruction system. We use a method called structure from motion to obtain object's sparse reconstruction and camera's extrinsic parameters. The latter are used for building a dense 3D model. Evaluation of extrinsic parameters shows that adding cameras to the reconstruction leads to accumulation of the error. Despite this occurrence the average error of the calculated poses remains within a few millimetres. Finally we evaluate the accuracy of the reconstructed models by comparing them with their ground truth. Results indicate small average errors, therefore we can conclude that the implemented system works sufficiently well.

Keywords: structure from motion, camera calibration, sparse reconstruction, dense reconstruction, 3D model.

Poglavje 1

Uvod

1.1 Motivacija

Kompleksni in relistični 3D modeli so uporabljeni na zelo različne načine. Nekatera vidnejša področja uporabe so animacija in oblikovanje, arhiviranje arheoloških najdb, spletni katalogi in nakupovanje obutve in oblačil. Zaradi potrebe po takšnih modelih, obstaja veliko kompleksnih orodij za 3D modeliranje, vendar je njihova uporaba zamudna in od uporabnika zahteva znanje in izkušnje. Takšen proces ročnega modeliranja je torej drag, zato so zanimive rešitve, ki objekt rekonstruirajo z malo napora in čim večjo natančnostjo. Ena izmed rešitev so 3D skenerji, ki so ponavadi zelo natančni. Obstaja veliko različnih principov na katerih ti skenerji delujejo, vsem pa je skupno to, da spadajo med opremo, ki je ponavadi draga in manj dostopna. Zaradi teh omejitev je zanimiv izziv na področju računalniškega vida rešitev s poudarkom na preprostosti in cenovni ugodnosti v zameno za nekaj natančnosti. Takšne rešitve objekt rekonstruirajo na podlagi zajetih fotografij, zato dodatne opreme ne potrebujejo, zadostuje že srednje zmogljiv računalnik in fotoaparat ali kamera. Na sliki 1.1 je prikazan model pridobljen s takšno vrsto rekonstrukcije.

Raziskovanje na področju 3D rekonstrukcije ima dolgo zgodovino, ki sega nazaj na področje fotogrametrije. Slednje je staro skoraj toliko kot moderna

Slika 1.1: Primer rekonstruiranega modela, ki se nahaja na arheološkem najdbišču v Turčiji.

fotografija in se ukvarja z natančnim merjenjem razdalj na podlagi zajetih fotografij. Glavno vprašanje pri rekonstrukciji, je kako iz slik pridobiti informacije o strukturi objekta. Problem je namreč nasproten zajemanju fotografije. Pri fotografiranju se točke objekta preslikajo na 2D površino, zato se informacija o globini izgubi, rekonstukcija pa skuša to informacijo pridobiti nazaj.

1.2 Sorodna dela

Metode za reševanje problema rekonstrukcije iz zajetih slik lahko razdelimo na aktive in pasivne. Pri aktivnih metodah so viri svetlobe nadzorovani, kar pomeni da poznamo njihovo postavitev ali vzorec, s to informacijo pa si pomagamo pri rekonstrukciji. Pri pasivnih metodah nimamo nadzora nad virom svetlobe, zato je vse informacije potrebno pridobiti direktno iz slik.

1.2.1 Aktivne metode

Omenili smo, da je pri tej vrsti rekonstrukcije vir svetlobe nadzorovan, v ta namen pa je pogosto uporabljen laser. Na sliki točko laserja zlahka detektiramo in nato rekonstruiramo, tako da poiščemo presečišče med laserskim

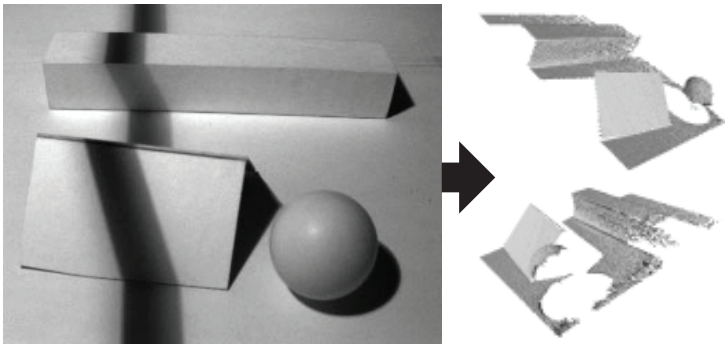
žarkom in navideznim žarkom, ki poteka skozi detektirano točko in središčem kamere. Če poznamo pozo laserja in kamere, potem lahko to storimo s triangulacijo. S takšnim postopkom je potrebno rekonstruirati vsako točko objekta posebej, kar pa v praktičnih aplikacijah ni zadovoljivo. Da bi se znebili potrebe po skeniranju objekta, moramo projecirati več točk naenkrat. Ta ideja se izkaže za problematično, ker pridela več možnih rešitev. V projekcijo je zato potrebno zakodirati informacijo, na podlagi katere lahko točke ločimo med seboj. Takšno projekcijo imenujemo strukturirana svetloba (angl. structured light).

Primer rekonstrukcije s strukturirano svetlobo je Microsoft Kinect [31]. Njegove glavne komponente so IR projektor, IR kamera in RGB kamera. Deluje tako, da projektor v prostor projecira gost vzorec točk, ki jih zazna IR kamera. Sistem je kalibriran, zato se točke lahko rekonstruirajo, po potrebi pa se jim lahko določi tudi barva z RGB kamere. Zhang idr. v svojem delu [28] na objekt projecirajo črtast vzorec izmenjajočih se barv. 3D oblika je pridobljena iz deformacije črt, različne barve pa poenostavijo iskanje ujemanj. Zanimiva je tudi metoda, ki je opisana v članku [7] in prikazana na sliki 1.2. V tem primeru za osvetlitev ni potrebna nobena dražja oprema, temveč samo namizna svetilka in svinčnik. Postopek poteka tako, da objekt osvetlimo, nato pa pred svetilko pomikamo svinčnik tako, da njegova senca preleti objekt. 3D oblika je pridobljena iz opazovane lokacije sence.

1.2.2 Pasivne metode

Pri tej vrsti rekonstrukcije nadzor nad osvetlitvijo ni potreben, zato so ponavadi dostopnejše in preprostejše za uporabo. Informacije o obliki so lahko pridobljene iz različnih iztočnic, kot na primer gradient tekstur, spremembe v ostrini slike, sence, prekrivanje objektov in še nekaterih drugih. Omenjene metode temeljijo na predpostavkah, ki v praksi pogosto niso izpolnjene, zato niso uporabljene tako pogosto. Bolj zanimive so rešitve ki temeljijo na rekonstrukciji iz večih pogledov.

Zanimiv primer takšne vrste je prostorsko klesanje (angl. space carving),

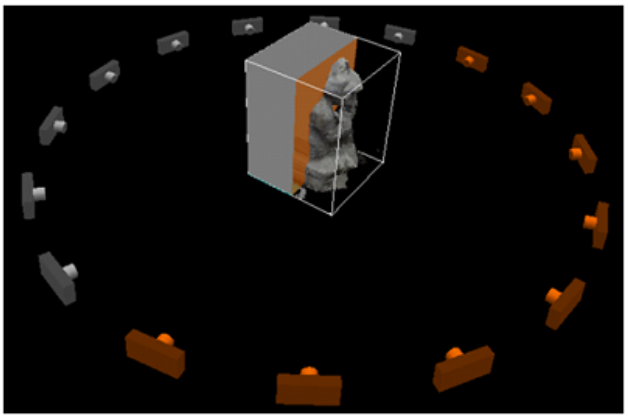


Slika 1.2: Preprost pristop za rekonstrukcijo na podlagi sence. Prikazan je primer rezultata iz članka [7].

ki je opisano v članku [17], njegovo delovanje pa je prikazano na sliki 1.3. 3D model objekta je pridobljen iz njegove silhuete na slikah, zato je pomemben korak njegova segmentacija. Rekonstrukcija se prične z gostim oblakom točk v obliki kvadra. Vsaka točka je projecirana na vse kamere na katerih je vidna. Točka je odstranjena, če se izkaže da ni del objekta, v nasprotnem primeru pa je ohranjena.

V praksi je pogosto uporabljen stereo sistem. To je sistem, kjer dve kameri istočasno zajameta sliko z dveh različnih zornih kotov. Ponavadi so v takšnem primeru poznani parametri kamer in njuna medsebojna lega, zato pravimo, da sta kameri kalibrirani. Najzahtevnejši del je iskanje ujemanj med točkami. Za vsako točko prve slike mora namreč biti poznana tudi lokacija iste točke na drugi sliki. Rekonstrukcija poteka preko triangulacije, kar pomeni da se poišče presečišče med navideznima žarkoma, ki ju tvorita ujemajoči se točki.

Bolj splošen primer takšnega sistema je postopek imenovan struktura iz gibanja (angl. structure from motion). V tem primeru je z eno kamero zajetih več slik statičnega objekta. Gibanje kamere je lahko poljubno, zato je poleg strukture potrebno tudi sočasno računanaje lege kamere za vsako izmed slik. Ko so lege kamer poznane, postane problem podoben stereo sistemu. Prednost takšnega postopka je možnost uporabe velikega števila slik, kar



Slika 1.3: Pri metodi prostorskega klesanja se rekonstrukcija prične z gostim oblakom točk, nato pa so odstranjene tiste, ki niso del objekta. Prikazan je rezultat iz članka [17].

omogoča samodejno kalibracijo in celovitejše rekonstrukcije. Te lastnosti naredijo strukturo iz gibanja zelo popularno rešitev. Naj še omenimo, da se v literaturi pogosto uporablja tudi pojem stereo večih pogledov (angl. Multi view stereo) in se nanaša na sistem kalibriranih kamer.

Eden izmed prvih pristopov, ki je lahko uvrščen med takšne metode je opisan v članku [24]. Avtorja v njem obravnavata problem strukture iz gibanja s pomočjo matrične faktorizacije, rešitev pa je poiskana z algoritmom, ki temelji na SVD razcepu. Glavna omejitev metode je, da predpostavlja ortografsko projekcijo. Pollefeys idr. v članku [23] predstavijo sodoben sistem za rekonstrukcijo. Od uporabnika se zahteva le vhodno zaporedje slik, sistem pa poskrbi za vse preostale korake, od samodejne kalibracije kamere do končne rekonstrukcije. Bolj podroben opis enake metode je podan v daljšem delu [21]. Po takšnem postopku je povzeta tudi naša rešitev z nekaterimi poenostavitvami in spremembami.

1.3 Cilji

V tem delu se bomo ukvarjali z rekonstrukcijo na podlagi zajetih fotografij. Zaradi svoje uporabnosti je to področje precej raziskano. Uporabili bomo postopek imenovan struktura iz gibanja. Njegovo poimenovanje se nanaša na sočasno računanje lege kamer in strukture objekta. Postopek je mogoče realizirati na različne načine kar pomeni, da so podrobnosti implementacije in uporabljeni algoritmi lahko zelo različni. V našem delu smo se odločili za tiste, ki so preprostejši za razumevanje in implementacijo, a vendar dovolj natančni in robustni.

Uporabljene algoritme bomo podrobno opisali, ter podali komentar glede alternativ. Postopek je bil realiziran v programskem jeziku Matlab in nekaterimi knjižnicami, ostale podrobnosti implementacije pa bomo podali v nadaljevanju. Hitrost izvajanja v našem primeru ni bila prioriteta.

1.4 Zgradba diplomske naloge

Delo je razdeljeno na šest poglavij. V Poglavju 2 opišemo teoretično ozadje, na katerem temeljijo uporabljeni algoritmi. V poglavju 3 so podrobno opisani algoritmi za iskanje značilnic, računanje osnovne matrike, robustno ujemanje značilnic, triangulacija, dodajanje kamere k obstoječi rekonstrukciji, globalna optimizacija, rektifikacija in računanje disparitete. Poglavje 4 na kratko opisuje postopek rekonstrukcije, ki je bil izbran. Podane so tudi utemeljitve o izbiri algoritmov, slike z vmesnimi rezultati in uporabljene knjižnjice ter ostala orodja. V poglavju 5 je predstavljen način eksperimentalnega vrednotenja. Podane so dobljene rekonstrukcije in njihova natančnost. Delo je zaključeno s poglavjem 6 kjer so povzeti rezultati in navedeni predlogi za izboljšave in nadaljnji razvoj.

Poglavje 2

Teoretična podlaga

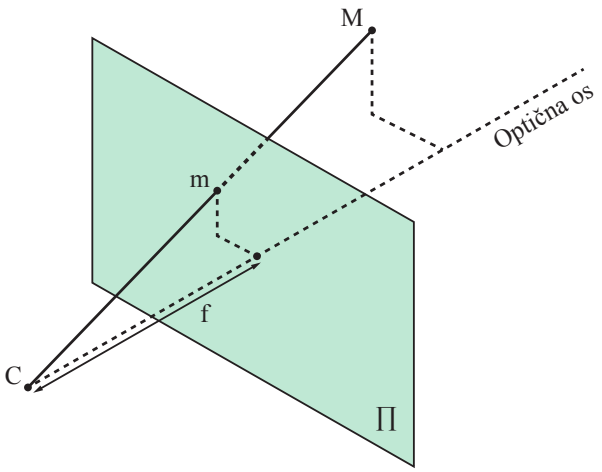
V tem poglabju bomo predstavili teorijo, ki stoji za uporabljenimi algoritmi. Opisali bomo model kamere in geometrijo, ki kamere med seboj povezuje.

2.1 Model kamere

Model kamere s točkasto odprtino je najpreprostejši model [8], ki opisuje preslikavo točk iz 3D prostora na 2D ravnino. V takšnem modelu se vsaka točka objekta na slikovno ravnino projecira samo preko enega žarka. To pomeni, da ni leče, ki zbira svetlobo, zaslonka kamere pa je predstavljena kot točka.

Opisani model preuredimo tako, da zamenjamo točkasto odprtino in slikovno ravnino in dobimo nov model, ki mu je ekvivalenten. Razlika je le v tem, da dobljena slika ni obrnjena in je zato enostavnejši za računanje. Takšno projekcijo si lahko predstavljamo tako, da se žarki odbiti od objekta napotijo proti centru projekcije, slika pa nastane na presečišču žarkov in slikovne ravnine (od centra projekcije je oddaljena za goriščno razdaljo). Slika 2.1 še grafično prikazuje opisani postopek.

Da takšen model lahko upošteva lastnosti resničnih kamer, ga je potrebno razširiti z dodatnimi parametri. Posledica majhnih napak pri izdelavi kamer je ta, da optična os ni natančno poravnana s središčem slikovne ravnine, zato



Slika 2.1: Projekcija 3D točke M na slikovno ravnino π po centralno projekcijskem modelu kamere. Slika je povzeta po [21].

potrebujemo dva nova parametra c_x in c_y, ki predstavljata omenjeni odmik. Poleg tega potrebujemo še dva parametra za goriščno razdaljo, in sicer f_x in f_y. Zanima nas namreč goriščna razdalja izmerjena v slikovnih elementih (angl. pixels), ki pa v kameri niso popolnoma kvadratni, kar je tudi posledica nenatančne proizvodnje. Omenili smo torej štiri parametre, imenujejo pa se notranji parametri kamere. Zapišemo jih v kalibracijski matriki K

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.1}$$

Kamera se nahaja nekje v trodimenzionalnem prostoru, zato ima še šest parametrov ki predstavljajo transformacijo iz koordinatnega sistema objekta v koordinatni sistem kamere. Od tega so trije potrebni za translacijo in trije za rotacijo (s pomočjo Rodriguesove formule lahko rotacijske parametre pretvorimo v rotacijsko matriko). Imenujemo jih zunanji parametri kamere

in jih zapišemo v matriki P

$$P = \Big[R \mid t\Big]; \quad R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad t = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}. \tag{2.2}$$

Celotno enačbo za projekcijo točke M iz 3D prostora v slikovni element m zapišemo z enačbo (točki sta zapisani v homogenih koordinatah)

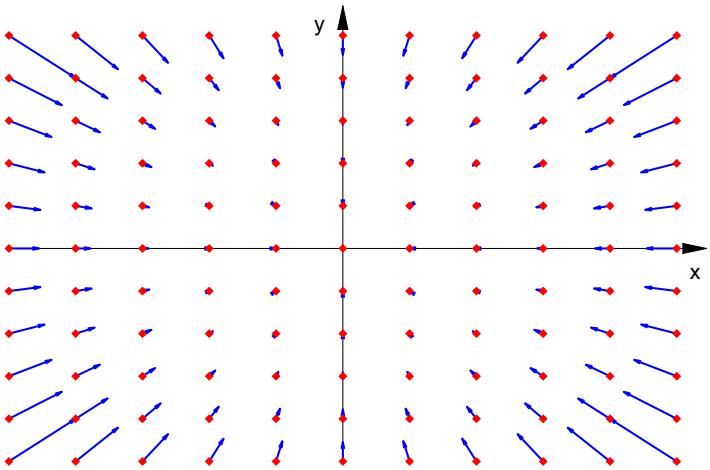
$$m = KPM; \quad m = \frac{1}{z}\begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.3}$$

Kamera ima tudi lečo, ki sliko nekoliko deformira. Ta učinek je prikazan na slikah 2.2 in 2.3. Dve glavni deformaciji sta radialna in tangencialna distorzija. Prva nastane zaradi okroglaste oblike leče, druga pa zato, ker leča in slikovna ravnina nista povsem paralelni. Matematično lahko definiramo lečo, ki slike ne deformira, ampak ta ni praktična za proizvodnjo. Radialna distorzija je dobro karakterizirana s tremi parametri (k_1, k_2, k_3) in jo modeliramo z (2.4), tangencialna pa z dvema (p_1, p_2) in jo modeliramo z (2.5). Modeliramo jo lahko tudi z drugačnim številom parametrov, ampak to ponavadi ni potrebno. Če so koeficienti poznani, potem lahko distorzijo odpravimo. To storimo tako, da najprej iz enačbe izpostavimo neznanki x in y. Dobimo enačbi s katerima lahko za vsak slikovni element pridobimo njegove popravljene koordinate. Pri odpravljanju distorzije je pomembna tudi interpolacija. Na novi sliki se lahko namreč pojavijo mesta, ki po preslikavi ostanejo prazna, zato jih zapolnimo z interpoliranimi vrednostmi.

$$\begin{aligned} x_{deformiran} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{deformiran} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{aligned} \tag{2.4}$$

$$\begin{aligned} x_{deformiran} &= x + (2p_1 xy + p_2(r^2 + 2x^2)) \\ y_{deformiran} &= y + (p_1(r^2 + 2y^2) + 2p_2 xy) \end{aligned} \tag{2.5}$$

$$\text{kjer je } r = \sqrt{(x^2 + y^2)}$$



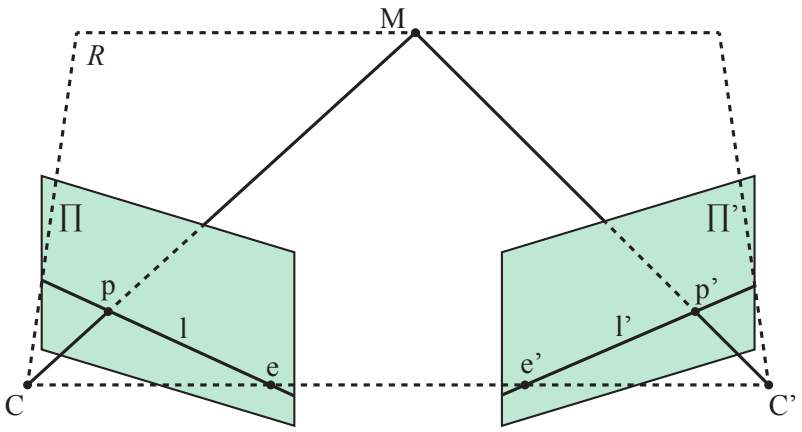
Slika 2.2: Kamera zaradi distorzije točke prestavi v smeri vektorjev, ki jih predstavljajo modre puščice. Opazimo lahko, da so pod večjim vplivom točke, ki so bolj oddaljene od središča.

Slika 2.3: Učinek radialne distorzije na konkretnem primeru.

Kamera ima torej skupno petnajst parametrov, ki jih lahko izračunamo s postopkom kalibracije, opisanim v poglavju 3.2. V primeru, da z isto kamero zajamemo več slik objekta, lahko predpostavimo da so notranji in distorzijski parametri novih slik isti, neznani pa ostanejo zunanji parametri.

2.2 Epipolarna geometrija

Epipolarna geometrija opisuje povezavo med točkami dveh slik. Na sliki 2.4 je prikazan sistem dveh kamer. Točka M se nahaja nekje v prostoru in je vidna v obeh kamerah. Njeni projekciji na slikovni ravnini Π in Π' sta točki p in p'. Ti točki nista predstavljeni v slikovnih elementih, ampak v fizičnih koordinatah, zato pravimo da sta normalizirani. Točko iz slikovnih elementov v fizične koordinate preslikamo tako, da jo pomnožimo z inverzom kalibracijske matrike. Premica skozi centra projekcije C in C' seka ravnini v epipolih e in e'. Lastnost takšnega sistema je, da za vsako točko m na levi sliki obstaja epipolarna premica l' na desni sliki, na kateri leži točka p'.



Slika 2.4: Sistem dveh kamer in epipolarna geometrija, ki opisuje razmerje med točkami na sliki. Slika je povzeta po [21].

Povezavo med točko in njeno epipolarno premico opisuje osnovna matrika E. Za njeno izpeljavo uporabimo epipolarno ravnino R, na kateri ležijo točka M in njeni projekciji, centra projekcije in oba epipola. To lastnost lahko zapišemo tudi s tremi koplanarnimi vektorji. Prvi vektor p definiramo kot točko na levi sliki. Drugi vektor t definiramo kot premik desne kamere glede na levo. Tretji vektor Rp' pa definiramo kot točko na desni sliki pomnoženo z rotacijo desne kamere glede na levo. Koplanarnost teh vektorjev zapišemo

z enačbo

$$p \cdot (t \times Rp') = 0. \tag{2.6}$$

Vektorski produkt lahko nadomestimo z množenjem matrik, tako da vektor t zapišemo v obliki 3×3 antisimetrične matrike $[t]_\times$. Upoštevamo še, da velja $E = [t]_\times R$ in pridemo do enačbe za epipolarno omejitev

$$p^T E p' = 0. \tag{2.7}$$

Pomen epipolarne omejitve je naslednji. Rezultat množenja prvih dveh členov $p^T E$ je epipolarna premica l', ki je zapisana v obliki vektorja. Ker hočemo da točka leži na epipolarni premici, mora biti rezultat množenja $l'p' = 0$.

Poleg osnovne matrike obstaja tudi fundamentalna matrika F. Matriki sta povezani preko enačbe

$$E = K^T F K. \tag{2.8}$$

Fundamentalna matrika poleg informacije o premiku in rotaciji vsebuje tudi informacije o notranjih parametrih. V našem primeru poznamo notranje parametre, zato govorimo o osnovni matiki.

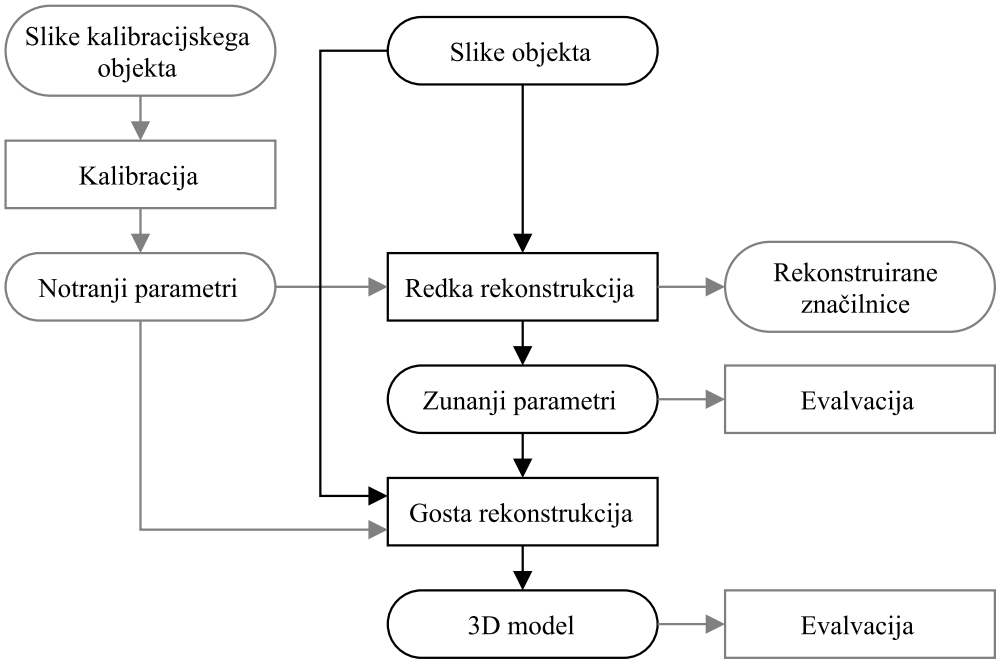
Poglavje 3

Metode in algoritmi

V tem poglavju bomo na začetku podali osnutek postopka rekonstrukcije, nato pa bomo podrobneje opisali algoritme, ki so bili uporabljeni kot gradniki za pridobitev 3D modelov.

3.1 Oris postopka

Naš postopek rekonstrukcije sestoji iz treh glavnih delov. Skupaj s svojimi vhodi in izhodi so prikazani na sliki 3.1. Prvi korak je kalibracija kamere, ki na podlagi zajetih slik kalibracijskega objekta poračuna notranje parametre kamere. Uporabimo jih v naslednjem koraku, ki mu pravimo redka rekonstrukcija. Vhod v ta del programa so poleg notranjih parametrov tudi slike objekta, ki ga želimo rekonstruirati. Rezultat so zunanji parametri kamer in redek oblak točk. Slednjega tvorijo rekonstruirane značilnice in nam v nadaljevanju ne koristi, zato ga lahko zavržemo. V zadnjem koraku, ki mu pravimo gosta rekonstrukcija, uporabimo informacijo o legi kamer, zato da za zaporedne pare kamer poračunamo globinsko sliko, iz katere pridobimo gost delni 3D model. Delne modele združimo v končno rekonstrukcijo.

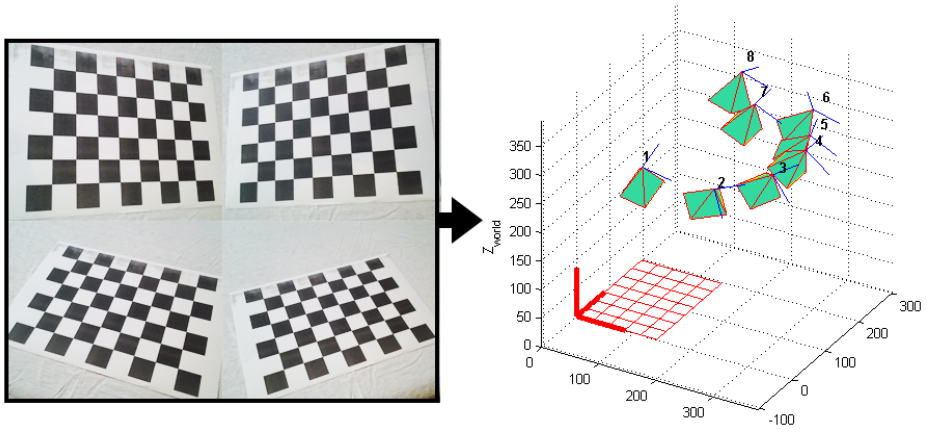


Slika 3.1: Oris postopka prikazuje glavne korake in njihove vhode oz. izhode.

3.2 Kalibracija kamere

S postopkom kalibracije, lahko pridobimo vse parametre kamere, za nadaljnje korake pri rekonstrukciji pa nas zanimajo notranji in distorzijski. Kamero lahko kalibriramo z različnimi tehnikami, te pa se delijo na dve glavni skupini, in sicer fotogrametrijska in samodejna kalibracija. Prva tehnika zahteva, da najprej zajamemo več slik kalibracijskega objekta in nato izračunamo parametre. Matematično gledano potrebujemo najmanj dve takšni sliki, v praksi pa za bolj kvalitetne rezultate uporabimo najmanj deset slik. Geometrija kalibracijskega objekta mora biti poznana čim bolj natančno (ponavadi je to vzorec šahovnice ali pik natisnjen na list papirja). Druga tehnika kalibracijskega objekta ne potrebuje. Notranje parametre namreč poračuna kar na podlagi slik uporabljenih pri gradnji modela.

V tem delu smo se odločili za fotogrametrijsko rešitev, ker je prepro-



Slika 3.2: Na levi strani je prikazanih nekaj slik uporabljenih za kalibracijo (kalibracijski objekt je tukaj šahovnica). Na desni pa je vizualizacija zunanjih parametrov kamer.

stejša in deluje bolj robustno. Slika 3.2 prikazuje rezultat takšne kalibracije, postopek pa je opisan v nadaljevanju.

3.2.1 Homografija

Izračun homografije za vsako izmed zajetih slik je prvi korak kalibracije. V računalniškem vidu predstavlja preslikavo ene ravnine v 3D prostoru v drugo (v našem primeru preslikava 2D kalibracijskega objekta v slikovno ravnino kamere). Ker gre za preslikavo ravnine lahko točko na objektu zapišemo kot $M' = [X, Y, 1]^T$. Preslikavo lahko sedaj zapišemo z enačbo (r_i je i-ti stolpec rotacijske matrike)

$$m = sHM'; \quad H = sK \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}, \quad m = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad M' = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{3.1}$$

Matrika H ima osem prostih parametrov zato jo lahko pomnožimo s poljubnim skalarjem. Normaliziramo jo s takšnim faktorjem s, da je $H_{33} = 1$. Če zgornjo enačbo primerjamo z (2.3) vidimo, da homografija zajema no-

tranje in zunanje parametre kamere, razlika pa je v tem da gre sedaj za preslikavo ravnine.

Za izračun homografije uporabimo algoritem imenovan DLT (ang. Direct linear transform) [15]. Naša predpostavka je, da poznamo geometrijo točk na kalibracijskem objektu M' in točke na sliki m. Najprej (3.1) zapišemo takole

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{3.2}$$

Ko elemente pomnožimo med seboj, dobimo enačbi

$$\begin{aligned} Xh_{11} + Yh_{12} + h_{13} - xXh_{31} - xYh_{32} - x = 0 \\ Xh_{21} + Yh_{22} + h_{23} - yXh_{31} - yYh_{32} - y = 0 \end{aligned} \tag{3.3}$$

Iščemo torej osem neznank, vsaka točka pa prispeva dve enačbi. Za oceno homografije zato potrebujemo najmanj štiri točke, zaradi šuma pa je bolje, če jih imamo več. Tak sistem enačb rešimo z metodo najmanjših kvadratov, zato enačbe in neznanke najprej zložimo v naslednjo obliko

$$Ah = 0;$$

$$A = \begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & 1 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_n \\ 0 & 0 & 0 & X_n & Y_n & 1 & -y_nX_n & -y_nY_n & -y_n \end{bmatrix}, h = \begin{bmatrix} h_{11} \\ h_{12} \\ h_{13} \\ \vdots \\ h_{33} \end{bmatrix}. \tag{3.4}$$

Matrika A v splošnem ni kvadratna (sistem je predoločen za $n > 4$ in nima rešitve), zato problem rešimo s pomočjo njenjega psevdo inverza A^TA, ki je kvadratna matrika. Rešitev sistema je lastni vektor, ki ustreza najmanjši lastni vrednosti A^TA. Vektor h učinkovito poiščemo s SVD razcepom (ang. Singular value decomposition), kot to prikazujeta enačbi

$$A \overset{svd}{=} UDV^T \tag{3.5}$$

$$h = \frac{[v_{19}, \ldots, v_{99}]^T}{v_{99}}. \tag{3.6}$$

3.2.2 Računanje notranjih parametrov

Opisali smo kako za vsako sliko kalibracijskega objekta pridobimo homografijo, sedaj pa lahko nadaljujemo z računanjem notranijih parametrov. Postopek je povzet po članku [30] in je podrobneje razložen v knjigi [8]. Zaenkrat predpostavimo, da slike nimajo distorzije. Reševanja se lotimo tako, da homografijo zapišimo v naslednji obliki

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \end{bmatrix} = sK \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}. \tag{3.7}$$

Kot smo že omenili, ima H osem prostih parametrov, zunanjih parametrov pa je šest, zato obstajata dve omejitvi na notranjih parametrih, ki ju predstavljata enačbi

$$\begin{aligned} r_1^T r_2 &= 0 \\ r_1^T r_1 &= r_2^T r_2. \end{aligned} \tag{3.8}$$

Izhajata iz lastnosti rotacijske matrike, in pomenita da sta vektorja r_1 in r_2 med seboj ortonormalna (pravokotna in enake dolžine). Iz (3.7) sledi

$$h_i = sKr_i \quad in \quad r_i = \frac{1}{s} K^{-1} h_i, \tag{3.9}$$

zato lahko (3.8) zapišemo tako

$$\begin{aligned} h_1^T K^{-T} K^{-1} h_2 &= 0 \\ h_1^T K^{-T} K^{-1} h_1 &= h_2^T K^{-T} K^{-1} h_2. \end{aligned} \tag{3.10}$$

Zanima nas matrika $K^{-T}K^{-1}$, ki jo bomo označili z B in je simetrična matrika. Zaradi te lastnosti lahko njene elemente zložimo v vektor b, enačbo pa preuredimo v obliko skalarnega produkta

$$h_i^T B h_j = \begin{bmatrix} h_{i1}h_{j1} \\ h_{i1}h_{j2} + h_{i2}h_{j1} \\ h_{i2}h_{j2} \\ h_{i3}h_{j1} + h_{i1}h_{j3} \\ h_{i3}h_{j2} + h_{i2}h_{j3} \\ h_{i3}h_{j3} \end{bmatrix}^T \begin{bmatrix} B_{11} \\ B_{12} \\ B_{22} \\ B_{13} \\ B_{23} \\ B_{33} \end{bmatrix} = v_{ij}^T b. \tag{3.11}$$

S takšno obliko pa sedaj začetni enačbi (3.8) zapišemo takole

$$\begin{gathered} Vb = 0 \\ \begin{bmatrix} v_{12}^T \\ v_{11}^T - v_{22}^T \end{bmatrix} b = 0. \end{gathered} \tag{3.12}$$

Vsi ti koraki so bili potrebni, da smo sistem enačb pretvorili v obliko, ki jo rešimo podobno kot pri računanju homografije. Enačbi, ki ju prispeva vsaka slika, zložimo v matriko V, rešitev pa dobimo s SVD razcepom. Notranje parametre pridobimo iz matrike B s spodnjimi enačbami.

$$\begin{aligned} f_x &= \sqrt{\lambda / B_{11}} \\ f_y &= \sqrt{(\lambda B_{11})/(B_{11}B_{22} - B_{11}^2)} \\ c_x &= (-B_{13}f_x^2)/\lambda \\ c_y &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \end{aligned} \tag{3.13}$$

kjer je

$$\lambda = B_{33} - (B_{13}^2 + c_y(B_{12}B_{13} - B_{11}B_{23}))/B_{11}$$

3.2.3 Računanje distorzijskih parametrov

Točke, ki se pojavijo na sliki v resnici niso na pravem mestu zaradi distorzije. Enačba (3.14) prikazuje to razmerje. Na levi strani enačbe imamo točko (X, Y, Z), ki se iz 3D koordinat preslika v kamero preko notranjih parametrov, ki so sedaj že znani. Na desni strani imamo točko (x, y), ki jo zaznamo na sliki, s še neznanimi parametri pa učinek distorzije izničimo (na podlagi (2.4) in (2.5)).

$$\begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} 2p_1 xy + p_2(r^2 + 2x^2) \\ p_1(r^2 + 2y^2) + 2p_2 xy \end{bmatrix} \tag{3.14}$$

Če zberemo dovolj takšnih enačb, lahko sistem rešimo in pridobimo distorzijske parametre. Notranje parametre je potrebno nato še enkrat poračunati, ker smo pri prvem računanju predpostavili, da distorzije ni (postopek ostane enak).

3.3 Značilnice

V računalniškem vidu je iskanje značilnic pomemben gradnik mnogih programov. Uporablja se na področjih razpoznavanja objektov, lokalizacije robota, šivanja panoram in 3D rekonstrukcije. Značilnice so točke na sliki z razpoznavno okolico, ki nam omogoča njihovo ponovljivo iskanje. To pomeni, da tudi če bi objekt bil slikan pod drugim kotom ali z drugačne razdalje, bi na sliki bile detektirane iste točke objekta. Takšnim točkam pripišemo še opisnik, ki ga izračunamo na podlagi slikovnih elementov v okolici in omogoča iskanje ujemanj med točkami.

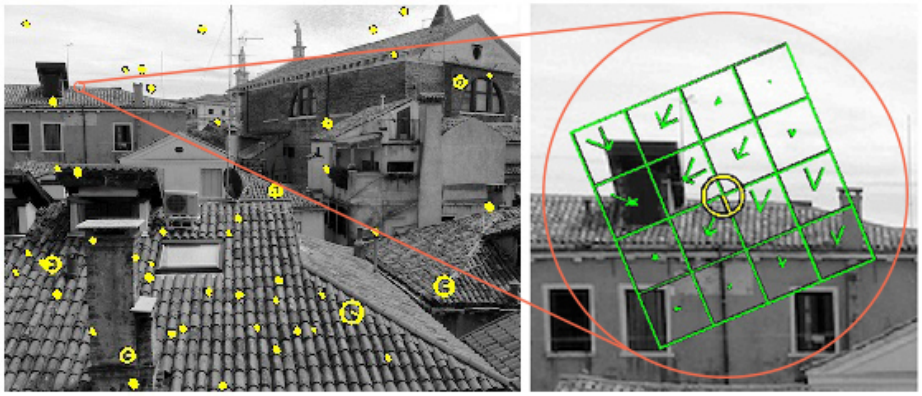
Primer značilnic so točke z močnim gradientom v različnih smereh, ki jim pravimo koti. Detektorjev značilnic je veliko, razlikujejo pa se v tem, da nekateri dajejo prednost hitrosti, drugi pa robustnosti. Nekateri izmed bolj popularnih so SIFT [20], SURF [4], BRISK [19] in FREAK [3]. V tem delu smo se odločili za uporabo detektorja SIFT. S postopki, ki so opisani v nadaljevanju, dosežemo, da so značilnice SIFT invariantne na lokacijo, velikost, rotacijo in spremembo osvetlitve. Leva stran slike 3.3 prikazuje značilnice, ki jih detektira algoritem.

3.3.1 Detekcija ekstremov

Prvi korak algoritma je identifikacija potencialnih lokacij in velikosti značilnic. Da lahko določimo njihovo velikost je ekstreme potrebno iskati v vseh možnih velikostih. SIFT za ta namen uporablja funkcijo DoG (angl. Difference of Gaussian), ki jo zapišemo s spodnjo enačbo. Funkciji G predstavljata Gaussovo krivuljo katerih parameter σ se razlikuje za faktor k.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y)$$

Postopek deluje tako, da sliko najprej večkrat gladimo z Gaussovim filtrom, pri tem pa enakomerno povečujemo parameter σ. Vmesne rezultate shranimo v zaporedje slik, od najbolj do najmanj ostre. Nato jih v zaporednih parih med seboj odštejemo in dobimo slike filtrirane z funkcijo DoG.



Slika 3.3: Na levi strani je prikazan rezultat algoritma SIFT. Prikazane so značilnice s svojo velikostjo in orientacijo. Na desni strani pa je vizualizacija enega izmed opisnikov. Okolica točke je razdeljena na 4x4 predelov, vsak predel pa ima svoj histogram orientacij. Obe sliki sta pridobljeni iz vadnice za knjižnico VLFeat [27].

Dobljeno zaporedje predstavlja velikosti slik (od najmanjše do največje). Velikost točke je torej odvisna od tega, na kateri sliki v zaporedju se nahaja. Lokalne ekstreme poiščemo tako, da točko primerjamo z vsemi sosedi (to so tudi točke na predhodnji in naslednji velikosti).

3.3.2 Lokalizacija točk

Vsaki detektirani točki najprej popravimo lokacijo, tako da s pomočjo njene okolice naredimo interpolacijo kvadratne funkcije in poiščemo ektrem na dobljeni krivulji. Rezultat je nova lokacija, ki je natančnejša in stabilnejša. Detekcija ekstremov pridela veliko točk med katerimi niso vse stabilne, zato se znebimo tistih ki imajo prenizko vrednost in tistih ki pripadajo robovom in ne kotom.

3.3.3 Določitev orientacije

Pri računanju orientacije uporabimo zaporedje glajenih slik iz prvega koraka. Lokacijo na glajeni sliki označimo z $L(x, y)$, stopnja glajenja pa je odvisna od velikosti točke. Za vsako sliko L poračunamo magnitudo in orientacijo z enačbama

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))).$$

Iz okolice posamezne točke in njihovih orientacij nato tvorimo histogram z 36 razdelki. Pri tem je vsak primerek utežen s svojo magnitudo in še dodatno Gaussovo krivuljo s središčem v točki sami (zato da ima bližnja okolica večji vpliv). Orientacijo, ki pripada največji vrednosti histograma pripišemo točki oz. značilnici. Za bolj robustno delovanje se lahko isti točki pripiše več orientacij. To storimo, če je v histogramu še kakšna vrednost dovolj blizu maksimuma.

3.3.4 Opisnik

Opisnik je pomemben zato, da lahko med značilnicami iščemo ujemanja. Njegovo računanje pričnemo podobno kot računanje orientacije. Iz glajenih slik namreč poračunamo magnitude in orientacije, tako kot smo to storili v prejšnjem koraku. Okolico posamezne značilnice (to je 16×16 slikovnih elementov in jo utežimo z Gaussovo krivuljo) razdelimo na 4×4 manjših predelov. V vsakem predelu nato tvorimo histogram orientacij z 8 razdelki. Opisnik je zato predstavljen kot vektor 128 elementov ($4 \times 4 \times 8$). Na koncu tak vektor še normaliziramo in s tem izboljšamo njegovo invariantnost na spremembo osvetlitve. Desna stran slike 3.3 grafično prikazuje takšen opisnik.

3.4 Osem-točkovni algoritem

To je eden izmed mnogih algoritmov za računanje osnovne matrike [29]. Zanj smo se odločili ker je preprost za implementacijo in dovolj zmogljiv za naše

potrebe. Vhod v algoritem so ujemajoče se točke dveh slik. Z istim algoritmom lahko računamo tudi fundamentalno matriko, tako da mu za vhod podamo nenormalizirane pare točk.

Vhodne točke predhodno transliramo tako, da je izhodišče v njihovem povprečju in skaliramo, da je njihova povprečna razdalja od središča $\sqrt{2}$. To storimo za vsako sliko posebej. Ta preprosta transformacija naredi algoritem precej bolj robusten, njegova zmoglivost pa postane primerljiva z bolj kompleksnimi rešitvami [14].

Če upoštevamo da sta $p = [x, y, 1]^T$ in $p' = [x', y', 1]^T$, potem lahko v (2.7) elemente pomnožimo med seboj. Neznanke osnovne matrike spravimo v ločen vektor in dobimo

$$\begin{bmatrix} xx' & xy' & x & yx' & yy' & y & x' & y' & 1 \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{12} \\ \vdots \\ E_{33} \end{bmatrix} = 0. \tag{3.15}$$

Fizične velikosti sistema ni mogoče določiti brez dodatnega referenčnega objekta, zato je skaliranje osnovne matrike poljubno. Imamo torej osem prostih parametrov, zato potrebujemo vsaj osem parov točk. Vsak par prispeva eno enačbo. Zložimo jih v skupni sistem enačb

$$Ae = 0;$$

$$\begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ x_2x_2' & x_2y_2' & x_2 & y_2x_2' & y_2y_2' & y_2 & x_2' & y_2' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_n' & x_ny_n' & x_n & y_nx_n' & y_ny_n' & y_n & x_n' & y_n' & 1 \end{bmatrix} \begin{bmatrix} E_{11} \\ E_{12} \\ \vdots \\ E_{33} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{3.16}$$

Rešitev sistema želimo dobiti po metodi najmanjših kvadratov. To storimo s SVD razcepom matrike A in enakim postopkom kot pri računanju homografije. Zaradi šuma v podatkih dobljena matrika $\hat{E}$ še ni prava osnovna matrika. Ta namreč zahteva, da sta njeni prvi dve lastni vrednosti enaki, tretja pa je enaka nič. To pomanjkljivost odpravimo tako, da ponovno uporabimo SVD razcep in dobimo (3.17). D je matrika lastnih vrednosti, zato

jo nadomestimo z matriko $\hat{D}$. Pravo osnovno matriko dobimo s (3.18).

$$\hat{E} \stackrel{svd}{=} UDV^T \tag{3.17}$$

$$E = U\hat{D}V^T, \quad \hat{D} = diag(1, 1, 0) \tag{3.18}$$

3.4.1 Lega kamere iz osnovne matrike

Izhodišče koordinatnega sistema si izberemo v levi kameri, zato je njena projekcijska matrika $P = [I|0]$. Zanima pa nas kakšna je lega desne kamere glede na levo. To informacijo pridobimo iz osnovne matrike. V splošnem obstajajo štiri možnosti za projekcijsko matriko druge kamere

$$\begin{aligned} P' = &\left[UWV^T \mid u_3\right] \text{ali} \\ &\left[UWV^T \mid -u_3\right] \text{ali} \\ &\left[UW^TV^T \mid u_3\right] \text{ali} \\ &\left[UW^TV^T \mid -u_3\right]. \end{aligned} \tag{3.19}$$

Matriki U in V sta rezultat SVD razcepa matrike E iz (3.17), vektor u_3 je tretji stolpec matrike U, matrika W pa je definirana kot

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.20}$$

Omenili smo že, da ima osnovna matrika samo osem prostih parametrov in zato fizična velikost sistema ni znana. Posledica tega je, da lahko vektor u_3, ki predstavlja translacijo, pomnožimo s poljubnim skalarjem. V našem primeru je normaliziran tako, da ima dolžino ena.

Teoretična podlaga in izpeljava zgornjih matrik je bolj podrobno razložena v knjigi [15]. Pravilno projekcijsko matriko izberemo tako, da rekonstruiramo eno izmed točk za vse štiri možnosti. Prava rešitev je tista za katero točka leži pred obema kamerama.

3.5 Robustno ujemanje značlnic

Osemtočkovni algoritem opisan v poglavju 3.4 je občutljiv na šum v podatkih, zato moramo poskbeti, da pri vhodnih parih ni nepravilnih ujemanj. Poleg tega so nepravilna ujemanja nezaželjena tudi v nadaljnjih korakih programa, zato se jih skušamo znebiti. To storimo z algoritmom RANSAC (angl. Random Sample Consensus) [10].

Algoritem deluje tako, da najprej poračunamo osnovno matriko na najmanjši množici parov, ki jo zahteva osem-točkovni algoritem. Glede na dobljen rezultat lahko pare točk ločimo med tiste, ki se strinjajo z rešitvijo in tiste, ki se ne. Za par točk pravimo, da se strinja z rešitvijo, če točki nista preveč oddaljeni od svojih pripadajočih epipolarnih premic (npr. ne več kot en slikovni element). Naša začetna rešitev zelo verjetno ne bo najboljša, zato postopek mnogokrat ponovimo, na koncu pa obdržimo rešitev z največjo podporo. Število ponovitev je odvisno od odstotka napačnih ujemanj (tega ne poznamo, lahko pa ga ocenimo). Pare, ki se z rešitvijo ne strinjajo, zavržemo, iz vseh preostalih pa še zadnjič poračunamo osnovno matriko. V algoritmu 1 je povzet opisan postopek.

Algoritem 1 RANSAC za računanje osnovne matrike

Vhod: Pari točk p_{zac}, p'_{zac}; napaka ϵ; št. iteracij n

Izhod: Osnovna matrika E; filtrirani pari točk p, p'

1: **for** 1 do n **do**
2: Izberi 8 naključnih parov
3: Izračunaj osnovno matriko $\hat{E}$
4: Preštej pare, ki podpirajo $\hat{E}$
5: **end for**
6: $E \leftarrow$ matrika z največjo podporo (izračunana še enkrat z vsemi pari)
7: $p, p' \leftarrow$ pari, ki podpirajo E

3.6 Triangulacija

Ko poznamo lego dveh kamer, lahko ujemajoče značilnice rekonstruiramo v 3D prostoru. Postopek se imenuje triangulacija in bi v idealnih razmerah potekal tako, da bi poiskali presečišče med premicama, ki ju definirata vektorja p in p'. Problem je torej v podatkih brez šuma trivialen, v realnih primerih pa se premici ne sekata, zato je iskano točko možno določiti na več načinov.

Eden izmed slabših načinov presečišče določi na središču daljice, ki je pravokotna na obe premici. Eden izmed boljših načinov poizkuša poiskati novi točki $\hat{p}$ in $\hat{p}'$, tako da ti upoštevata epipolarno omejitev in sta čim bližje izvirnim točkam p in p'. Obstaja še več algoritmov za triangulacijo, z različnimi prednostmi in slabostmi [16]. V tem delu smo se odločili za postopek, ki minimizira reprojekcijsko napako. Ta pridela zadovoljive rezultate in je preprost za implementacijo. Njegova prednost je tudi ta, da lahko za rekonstrukcijo uporabimo več kot dva pogleda. Reprojekcijska napaka je definirana kot razdalja med detekcijami točke p_i in rekonstruirano točko projecirano nazaj v kamere P_iM.

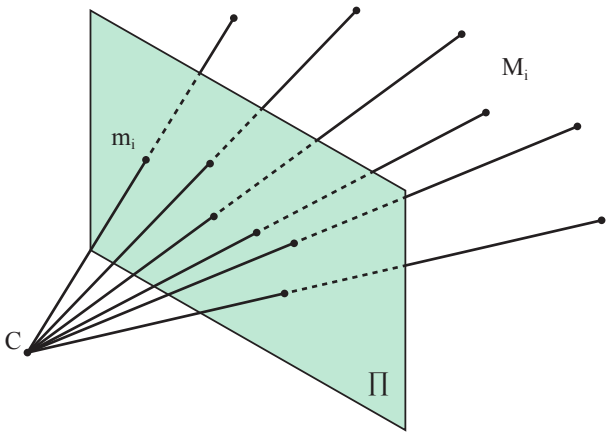
Točko poiščemo tako, da rešimo sistem enačb

$$AM = 0;$$
$$\begin{bmatrix} [p^1]_\times P_1 \\ [p^2]_\times P_2 \\ \vdots \\ [p^n]_\times P_n \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}. \tag{3.21}$$

Geometrično gledano zgornja enačba zahteva, da sta vektorja p^i in P_iM vzporedna, kar pomeni, da mora biti njun vektorski produkt enak nič. Rešitev takšnega sistema znova poiščemo s SVD razcepom in enakim postopkom opisanem pri računanju homografije.

3.7 Dodajanje nove kamere

Do sedaj smo s pomočjo dveh pogledov vzpostavili začetno rekonstrukcijo, vključiti pa želimo še ostale poglede. Poznavanje osnovne matrike nam v tem primeru ne pomaga. V poglavju 3.4.1 smo namreč omenili, da iz nje lahko pridobimo rotacijo in smer translacije, magnituda translacije pa ostane neznana.



Slika 3.4: Poznane so točke M_i in m_i, iščemo pa lego kamere C. DLT za izračun projekcijske matrike kamere potrebuje najmanj šest parov točk.

Projekcijsko matriko izračunamo z algoritmom za oceno lege kamere (angl. Pose estimation). Ta algoritem orientacijo in translacijo kamere izračuna na podlagi 3D točk M_i in njihovih 2D projekcij m_i na slikovno ravnino, kot to prikazuje slika 3.4. Podati moramo torej tiste značilnice iz nove slike, ki imajo ujemanja na prejšnjih slikah in so že rekonstruirane. Algoritem deluje tako, da za začeten približek uporabi DLT, nato pa z iterativnim postopkom rešitev izboljša. Obstajajo tudi metode, ki za delovanje potrebujejo manj točk in so hitrejše (npr. P3P [11]), ampak se v določenih primerih izkažejo za manj robustne.

Preslikavo 3D točk na slikovno ravnino opisuje (2.3). Ker poznamo

notranje parametre K lahko točke m nomaliziramo in dobimo

$$p = \begin{bmatrix} R \mid t \end{bmatrix} M; \quad p = \frac{1}{z} \begin{bmatrix} x \\ y \\ z \end{bmatrix}, \quad M = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.22}$$

Nato lahko koordinate točke p izrazimo z

$$\begin{aligned} x &= \frac{r_{11}X + r_{12}Y + r_{13}Z + t_x}{r_{31}X + r_{32}Y + r_{33}Z + t_z}, \\ y &= \frac{r_{21}X + r_{22}Y + r_{23}Z + t_y}{r_{31}X + r_{32}Y + r_{33}Z + t_z}. \end{aligned} \tag{3.23}$$

Vsaka točka prispeva dve enačbi, matrika P pa ima enajst neodvisnih spremenljivk. Potrebujemo torej vsaj šest točk in njihovih projekcij. Zgornji enačbi preoblikujemo tako, da so vsi členi na levi strani in jih spravimo v spodnji sistem enačb, ki ga rešimo s SVD razcepom.

$$Ax = 0; \tag{3.24}$$

$$A = \begin{bmatrix} X_1 & Y_1 & Z_1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z & 1 & 0 & -x_1 \\ 0 & 0 & 0 & X_1 & Y_1 & Z_1 & -y_1X_1 & -y_1Y_1 & -y_1Z & 0 & 1 & -y_1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ & 1 & 0 & -x_n \\ 0 & 0 & 0 & X_n & Y_n & Z_n & -y_nX_n & -y_nY_n & -y_nZ & 0 & 1 & -y_n \end{bmatrix},$$

$$x = \begin{bmatrix} r_{11} & r_{12} & r_{13} & r_{21} & r_{22} & r_{23} & r_{31} & r_{32} & r_{33} & t_x & t_y & t_z \end{bmatrix}^T.$$

Sistem enačb je podoben kot tisti pri računanju homografije. Razlika je v tem, da se pri homografiji predpostavlja, da točke ležijo na isti ravnini, tukaj pa te omejitve ni. Dobljena rešitev se uporabi kot začetni približek za nelinearno optimizacijo. Ta iterativni postopek skuša zmanjšati reprojekcijsko napako podanih točk in ga zapišemo z

$$\min_P \sum_{i=1}^{n} \left\| p_i - PM_i \right\|^2 \tag{3.25}$$

Algoritem postane nenatančen, če so med podanimi točkami tudi tiste, ki so rezultat nepravilnih ujemanj. V poglavju 3.5 smo opisali kako se znebimo večine takšnih parov, ampak ponavadi ne vseh, zato tudi ta algoritem uporabimo v kombinaciji z algoritmom RANSAC.

3.8 Globalna optimizacija

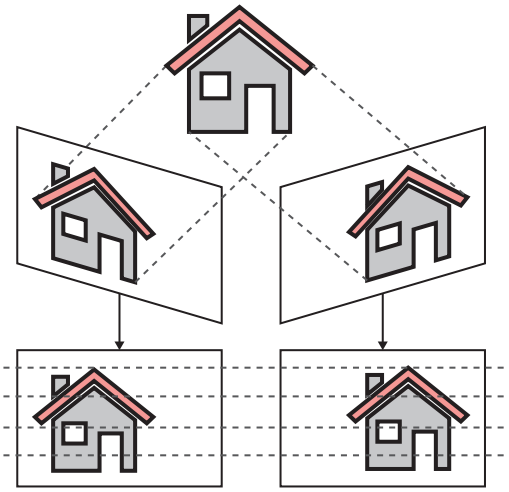
Natančnost rekonstruiranih točk in lege kamer lahko izboljšamo z globalno minimizacijo reprojekcijske napake (angl. Bundle adjustment). Poleg tega tudi zmanjšamo akumulacijo napake, ki je posledica zaporednega dodajanja kamer. Vhod v algoritem je začetni približek za zunanje parametre kamer in lokacije točk. Te vrednosti lahko pridobimo z do sedaj opisanimi postopki. Cilj algoritma je, da poiščemo nove zunanje parametre kamer P_j in 3D točke M_i, tako da je skupna reprojekcijska napaka čim manjša. Za notranje parametre predpostavimo, da so bili s postopkom kalibracije pridobljeni dovolj robustno, zato jih ne popravljamo in pri računanju uporabimo normalizirane koordinate točk. Omenili smo že, da je reprojekcijska napaka definirana kot razdalja med točko i na kameri j (zapisano s p_i^j) in projekcijo 3D točke i v kamero j (zapisano s $P_j M_i$). Naj še omenimo, da ni potrebno, da je posamezna točka vidna v vseh kamerah. Minimizacijo lahko zapišemo z

$$\min_{P_j, M_i} \sum_{j=1}^{k} \sum_{i=1}^{n} \left\| p_i^j - P_j M_i \right\|^2 . \tag{3.26}$$

Podani problem je zaradi velikega števila spremenljivk zelo težak, čeprav je izražen na preprost način. Vsaka kamera prispeva šest spremenljivk, vsaka točka pa tri. To pomeni, da imamo že pri manjših rekonstrukcijah več tisoč spremenljivk. Problem lahko rešimo veliko bolj učinkovito če upoštevamo posebno strukturo problema. Posamezne napake so namreč odvisne samo od dveh parametrov (točke M_i in kamere P_j). Posledično so matrike pri računanju odvodov zelo redke, zato lahko število spremenljivk, ki jih ocenjujemo naenkrat precej zmanjšamo. Algoritem ima še mnoge druge pomembne podrobnosti in je opisan v delu [25].

3.9 Rektifikacija

Če poznamo notranje in zunanje parametre dveh kamer, potem lahko na podlagi pripadajočih slik izračunamo dispariteto, in sicer tako, da poiščemo ujemanja za vsak slikovni element. Rektifikacija močno poenostavi računanje disparitete, saj iskanje ujemanj v 2D prostoru spremeni v iskanje vzdolž horizontalne premice. Rezultat rektifikacije sta namreč sliki z vzporednimi in poravnanimi epipolarnimi premicami, kot to prikazuje slika 3.5. Postopek poišče novo slikovno ravnino, ki je skupna obema kamerama. Teoretično je takšnih ravnin neskončno, zato se skuša izbrati tista, ki minimizira distorzijo slik.



Slika 3.5: Prikazana je skica rektifikacije. Spodnji sliki sta poravnani po vrsticah, zato je iskanje ujemanj poenostavljeno.

Uporabljena metoda izvira iz [6] in je podrobno opisana v knjigi [8]. Vhod v algoritem sta rotacijska matrika R in translacijski vektor t, ki transformirata desno kamero v koordinatni sistem leve. Matriko R razcepimo na dve polovični rotaciji r_l in r_r. Ta delitev zmanjša distorzijo dobljenih slik. Z dobljenimi matrikami rotiramo obe kameri tako, da njuni optični osi postaneta vzporedni. Slikovni ravnini kamer postaneta s to rotacijo vzporedni, vrstice

slik pa ostanejo neporavnane. V ta namen definiramo še eno rotacijsko matriko R_{rect}, ki epipol leve slike preslika v neskončnost in posledično poravna epipolarne premice. Definiramo jo s pomočjo spodnjih vektorjev.

$$\begin{aligned} e_1 &= \frac{t}{\| t \|} \\ e_2 &= \frac{[-t_y \; t_x \; 0]^T}{\sqrt{t_x^2 + t_y^2}} \\ e_3 &= e_1 \times e_2 \end{aligned} \tag{3.27}$$

Vektor e_1 definiramo v smeri epipola in ga normaliziramo. Ta smer je enaka translaciji. Vektor e_2 definiramo tako, da je pravokoten na e_1 in optično os kamere, kar dosežemo z vektorskim produktom. Tudi ta vektor normaliziramo. Vektor e_3 izračunamo z vektorskim produktom med e_1 in e_2. Zložimo jih v rotacijsko matriko

$$R_{rect} = \begin{bmatrix} (e_1)^T \\ (e_2)^T \\ (e_3)^T \end{bmatrix}. \tag{3.28}$$

Kameri rektificiramo s spodnjima enačbama. Predstavljata rotacijo, ki kameri transformirata tako, da sta vrstično poravnani, njuni slikovni ravnini pa sta koplanarni.

$$\begin{aligned} R_l &= R_{rect} r_l \\ R_r &= R_{rect} r_r \end{aligned} \tag{3.29}$$

3.10 Dispariteta

Na podlagi dveh rektificiranih slik, lahko poračunamo njuno globinsko sliko oziroma dispariteto. Z njeno pomočjo za vsak slikovni element leve slike pridobimo globino na kateri se nahaja. To storimo tako, da za slikovne elemente na levi sliki poiščemo najboljše ujemanje na desni sliki. Ker sta sliki vrstično poravnani, poteka iskanje v 1D prostoru. Dispariteto točke določimo tako, da odštejemo koordinato x desne točke od koordinate x leve

točke, kot to opisuje spodnja enačba.

$$d = x^l - x^r \tag{3.30}$$

Algoritem poteka v treh korakih. Najprej skušamo iz slik odpraviti razlike v intenzivnosti osvetlitve in poudariti teksturo. To storimo tako, da od vsakega slikovnega elementa odštejemo povprečno vrednost njegove okolice. Ta vrednost ima zgornjo mejo I_{max}, ki jo lahko določi uporabnik. V drugem koraku poiščemo ujemanja z drsečim oknom SAD (angl. Sum of absolute difference), kar pomeni da iščemo takšno točko, ki ima razliko v okolici čim manjšo. Parametra d_{min} in število disparitet določata spodnjo in zgornjo mejo disparitete. Posredno določita tudi razpon globine, ki jo algoritem lahko zazna. V zadnjem koraku se skušamo znebiti napačnih in nezanesljivih ujemanj. Obdržimo le tista, ki dovolj odstopajo od najslabšega kandidata. Do problemov pride tudi na robovih objekta, kjer se v oknu SAD na eni strani nahaja ospredje, na drugi pa ozadje. Posledica so šumna območja velikih in majhnih disparitet. Znebimo se jih tako, da dobljeno globinsko sliko filtriramo z oknom, ki zahteva, da znotraj njega ni prevelikih razlik v dispariteti.

3.10.1 Projekcija točk v 3D prostor

S pomočjo disparitete in notranjih parametrov rektificiranih kamer, lahko točke na sliki projeciramo v 3D prostor. V ta namen najprej definiramo matriko Q. Vsi notranji parametri se nanašajo na levo kamero, razen c'_x. Ta pride v poštev samo, če se optični osi kamer ne sekata v neskončnosti. V nasprotnem primeru je enak c_x, zato je posledično celoten člen enak nič. Razdalja med kamerama je označena z t_x.

$$Q = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & 1 & 0 & -c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & -1/t_x & (c_x - c'_x)/t_x \end{bmatrix} \tag{3.31}$$

Točko m na sliki s koordinatama x in y, ter dispariteto d projeciramo v prostor tako, da jo pomnožimo z matriko Q. Dobljena 3D točka M je predstavljena v homogenih koordinatah.

$$M = Q \begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{3.32}$$

Če to transformacijo naredimo za vse točke na sliki dobimo gosto rekonstrukcijo za en pogled.

Poglavje 4

Implementacija rešitve

V tem poglavju bomo opisali kako so predstavljeni algoritmi uporabljeni v našem programu in podali nekaj vmesnih rezultatov.

4.1 Postopek rekonstrukcije

Vhod v program so slike objekta, ki morajo biti zajete tako, da kamero premikamo samo v eno smer in da med zaporednimi slikami ne naredimo prevelikega premika. Za prikaz delovanja bomo v tem poglavju uporabili fotografije prikazane na sliki 4.1.

Slika 4.1: Devet slik uporabljenih za rekonstrukcijo.

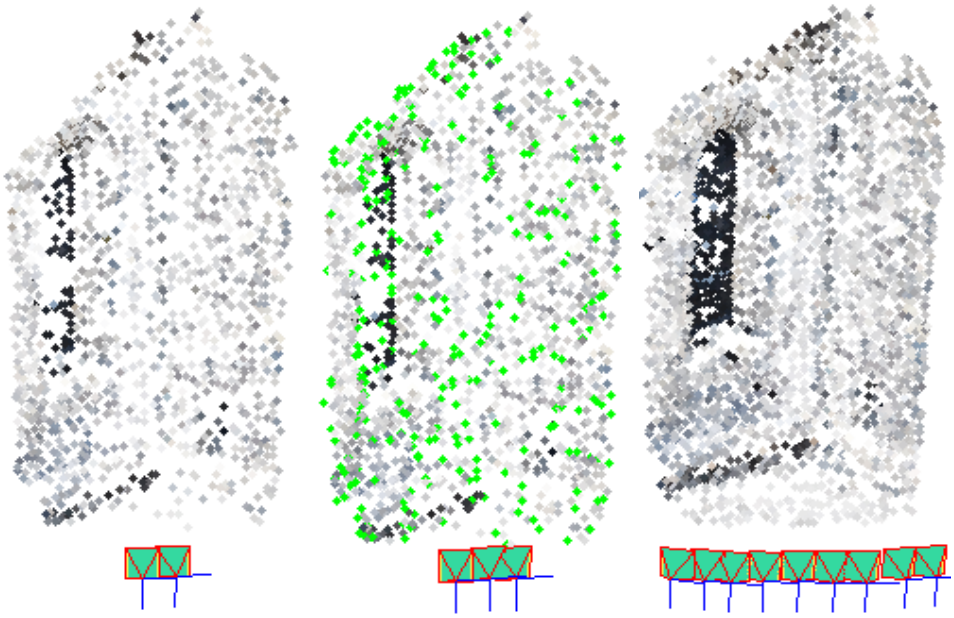
Naš postopek predpostavlja, da poznamo tudi notranje parametre kamere. Pridobimo jih lahko s kalibracijo kamere. V tem delu smo se odločili za ročno kalibracijo, ta je namreč preprostejša za implementacijo in ponavadi bolj robustna od avtomatične. Ročna kalibracija zahteva, da zajamemo nekaj

slik kalibracijskega objekta. V poglavju 3.2 smo omenili, da je za dober rezultat zaželjeno, da zajamemo vsaj deset takšnih slik. Rezultat kalibracije so notranji in distorzijski parametri kamere. Uporabljeni model obravnava distorzijo ločeno, zato je pomembno da jo na slikah odpravimo preden začnemo z rekonstrukcijo. To storimo s pomočjo poračunanih parametrov.

4.1.1 Redka rekonstrukcija

Rezultat tega koraka so lege kamer in rekonstruirane značilnice. Pričnemo tako, da poiščemo ujemanja med točkami dveh zaporednih pogledov. Problem je prezahteven, da bi to storili za vse slikovne elemente, zato z algoritmom SIFT na slikah poiščemo značilnice. Med zaporednimi pari slik pa nato poiščemo ujemanja med značilnicami. Ker še nimamo informacij o legi kamer, jih moramo poiskati tako, da izmed vseh možnih parov izberemo najboljše. Takšno naivno iskanje pridela tudi nekaj nepravilnih ujemanj. Znebimo se jih tako, da uporabimo RANSAC v kombinaciji z osem-točkovnim algoritmom. Slednji s pomočjo ujemanj poišče povezavo med dvema pogledoma. Njegov rezultat je osnovna matrika na podlagi katere lahko poračunamo medsebojno lego kamer. Ta korak je najpočasnejši del programa, zaradi iskanja značilnic in njihovih ujemanj.

Rekonstrukcijo vzpostavimo s prvima dvema pogledoma. Izhodišče koordinatnega sistema si izberemo v projekcijskem središču prve kamere. Rotacijo in premik druge kamere glede na prvo pa pridobimo iz osnovne matrike. Na ta način določimo lego obema kamerama, zato lahko ujemajoče se značilnice rekonstruiramo s triangulacijo. Postopek nadaljujemo z dodajanjem preostalih pogledov k rekonstrukciji. Kot smo to že omenili, nam poznavanje osnovne matrike pri dodajanju pogleda ne pomaga. Translacija pridobljena iz osnovne matrike je namreč lahko poljubno skalirana, kar pomeni da lega nove kamere glede na prvo kamero ne more biti enolično določena. Nov pogled k rekonstrukciji dodamo z algoritmom za oceno lege kamere. Značilnice detektirane na novi sliki lahko razdelimo na tiste, ki so že rekonstruirane in tiste ki še niso. Rekonstruirane značilnice in njihove pripadajoče 3D točke

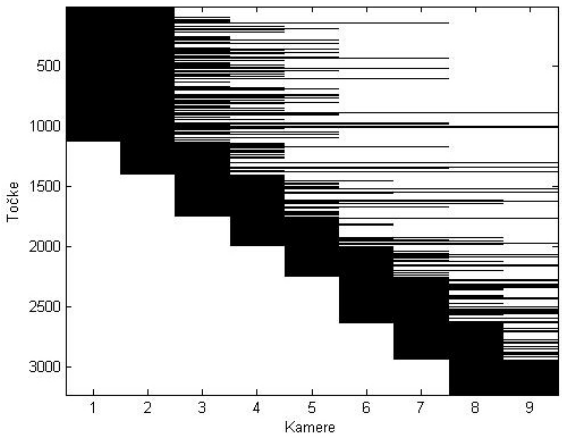


(a) (b) (c)

Slika 4.2: (a) Vzpostavitev rekonstrukcije s prvima dvema pogledoma. (b) Dodajanje novega pogleda k rekonstrukciji. Dodane točke so obarvane zeleno. (c) Rezultat redke rekonstrukcije.

uporabimo pri računanju lege nove kamere. Ker so v tem koraku še vedno lahko prisotna napačna ujemanja, ponovno uporabimo RANSAC, tokrat v kombinaciji z algoritmom za oceno lege kamere. Ko je nova kamera dodana, popravimo lokacijo obstoječih točk in k rekonstrukciji dodamo tudi tiste značilnice z nove slike, ki še niso rekonstruirane. Takšen postopek lahko privede do akumulacije napake, zato moramo po vsakem dodanem pogledu izvesti tudi globalno minimizacijo reprojekcijske napake. Postopek z nekaterimi vmesnimi korami je prikazan na sliki 4.2.

Sproti gradimo tudi masko, ki za vsako izmed 3D točk prikazuje na katerih kamerah je vidna. Na sliki 4.3 je prikazan primer takšne maske. Iz nje lahko razberemo, katere značilnice posamezne slike so že rekonstruirane in katere še niso. Uporabimo jo tudi pri globalni minimizaciji in določanju barv za vizualizacijo. Pri velikem številu pogledov in točk je tudi ta del



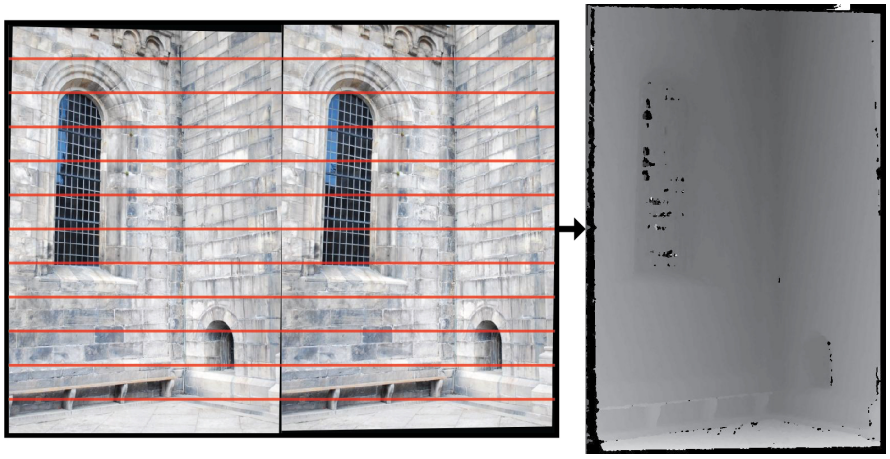
Slika 4.3: Maska za vsako točko prikazuje na katerih slikah je vidna.

programa lahko počasen. Dobljena rekonstrukcija sestoji iz nekaj tisoč točk in ji pravimo redka rekonstrukcija. Glavni rezultat so poračunane lege kamer, ki jih uporabimo pri gosti rekonstrukciji.

4.1.2 Gosta rekonstrukcija

Postopek nadaljujemo z gosto rekonstrukcijo, ki uporabi informacijo o legi kamer. Redek oblak rekonstruiranih značilnic nam v tem koraku ne pomaga, zato ga lahko zavržemo (prav bi nam prišel pri kakšnem drugem problemu, naprimer lokalizaciji). Končni model pridobimo z združevanjem delnih modelov. V našem primeru delne modele pridobimo na podlagi dveh kamer. Ker so kamere kalibrirane, lahko problem obravnavamo kot stereo sistem. Obstajajo tudi algoritmi za gosto rekonstrukcijo na podlagi večih pogledov, vendar jih v tem delu zaradi njihove kompleksnosti nismo uporabili.

Gosto rekonstrukcijo vzpostavimo z dvema pogledoma. Iskanje ujemanj lahko tokrat poenostavimo, ker poznamo legi kamer. To storimo z rektifikacijo. Rezultat sta vrstično poravnani sliki, zato je iskanje ujemanj enodimenzionalno. Takšno iskanje je hitro, zato lahko globino poiščemo za vsak slikovni element. Naj še omenimo, da je globina poiskana za slikovne elemente leve kamere. Na podlagi dobljene disparitete, lahko točke na sliki



Slika 4.4: Na levi je rektificiran par slik. Rdeče črte ponazarjajo njuno vrstično poravnanost. Dobljena dispariteta je prikazana na desni strani.

projeciramo v 3D prostor. Model moramo še transformirati nazaj v koordinatni sistem prve kamere. Na sliki 4.4 je prikazan rezultat poravnave in primer disparitete.

Pri dodajanju novega delnega modela (pridobljenega iz drugega para kamer) ne želimo dodajati že obstoječih točk. Temu se izognemo z masko, ki nam pove katere točke na levi sliki novega para so že rekonstruirane. Masko izračunamo tako, da obstoječe točke projeciramo na novo kamero. To nažalost ni popolnoma pravilna rešitev, saj se na kamero projecirajo tudi točke, ki jih nova kamera gleda od zadaj in v resnici na njej niso vidne. Rešitev izboljšamo tako, da točkam predhodno določimo normale. Na ta način vemo kam so točke usmerjene, njihovo vidnost na novi kameri pa preverimo s skalarnim produktom. Masko izboljšamo še z morfološko operacijo zapiranja, ki zapolne nekatere manjše luknje v maski. Rezultat goste rekonstrukcije je prikazan na sliki 4.5.

Združevanje delnih modelov zahteva več uporabniške interakcije. Pri računanju disparitete je potrebno nastaviti nekaj parametrov, ki ne delujejo vedno enako dobro za različne pare slik.

Slika 4.5: Rezultat goste rekonstrukcije.

4.2 Orodja in knjižnice

Postopek smo implementirali v razvojnem okolju Matlab in s pomočjo nekaterih knjižnic. Za kalibracijo, oceno lege kamere, rektifikacijo in računanje disparitete smo uporabili mexopencv [1]. Ta knjižnica nudi funkcije MEX kot vmesnik za OpenCV-jev aplikacijski programski vmesnik. OpenCV ponuja širok spekter algoritmov računalniškega vida, njegova implementacija pa je osredotočena na učinkovitost in delovanje v realnem času. Za iskanje značilnic smo uporabili VLFeat [26]. Poleg implementacije SIFT, ponuja tudi druge algoritme osredotočene na pridobivanje značilnic in iskanje njihovih ujemanj. Za globalno minimizacijo reprojekcijske napake smo uporabili knjižnico VLG [18]. Rekonstruirane modele smo zapisali v formatu ply. Za pisanje in branje teh datotek smo uporabili funkcije PLY_IO [13]. Za prikazovanje gostih modelov smo uporabili program MeshLab [2], ki poleg tega ponuja širok spekter algoritmov za obdelavo 3D modelov. Uporabili smo ga tudi pri računanju normal za točke delnih modelov. Dodajanje pogledov je zaradi tega vmesnega koraka zamudno.

Poglavje 5

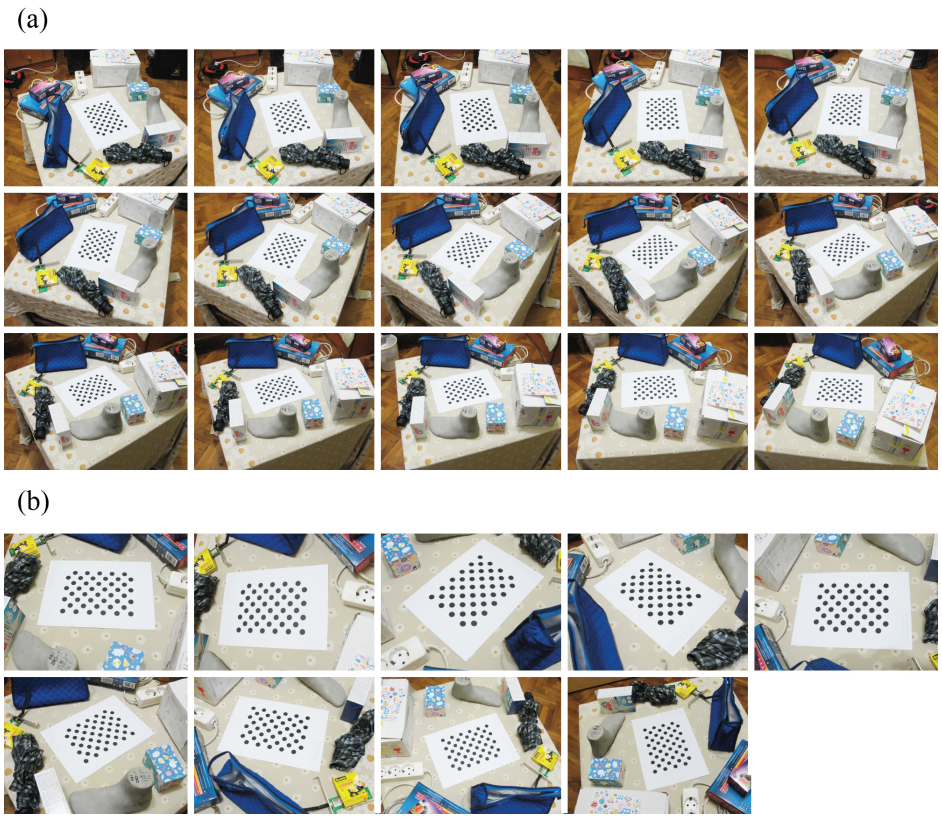
Eksperimentalno vrednotenje

Evalvacijo smo razdelili na tri dele. Najprej je opisana evalvacija zunanjih parametrov kamer. Nadaljujemo s kvantitativno evalvacijo rekonstrukcij, na koncu pa podamo še nekaj primerov rekonstrukcij.

5.1 Evalvacija zunanjih parametrov

Natančnost zunanjih parametrov smo evalvirali tako, da smo postopek redke rekonstrukcije izvedli nad slikami uporabljenimi za kalibracijo. Rezultat redke rekonstrukcije so lege kamer, njihove referenčne lege pa smo pridobili s postopkom kalibracije, ki je opisan v poglavju 3.2. Pridobljene zunanje parametre smo primerjali z referenčnimi legami in tako dobili napake v rotaciji in translaciji kamer.

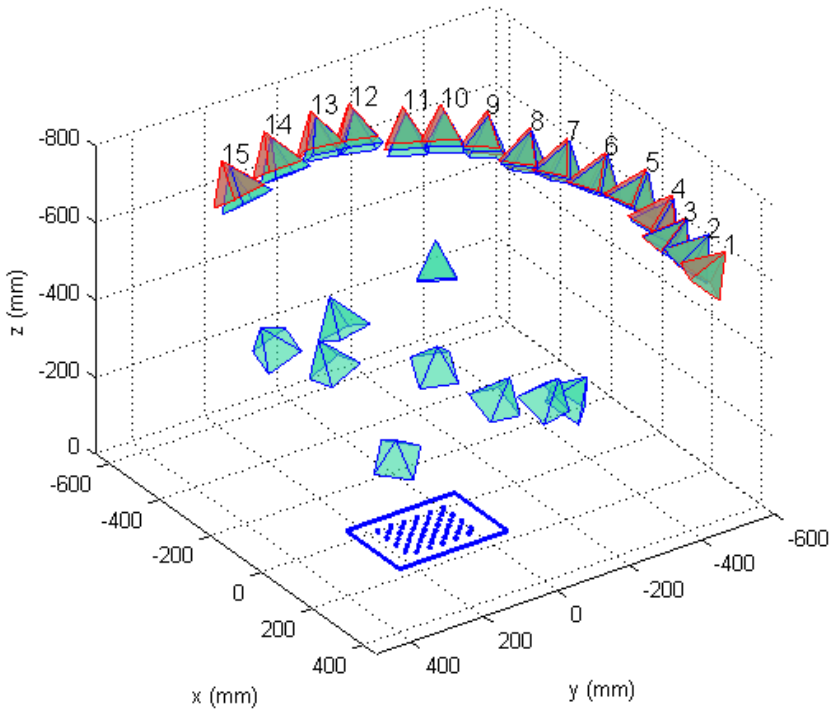
Kalibracijske slike smo morali zajeti na malce drugačen način. Pri rekonstrukciji namreč pride do težav, če model nima teksture in je preveč ploščat. Ta problem smo rešili tako, da smo okoli kalibracijskega objekta postavili razne teksturirane predmete. Prvih 15 slik smo zajeli tako, da smo kamero premikali v eno smer z majhnimi premiki. Te so primerne za rekonstrukcijo, za potrebe kalibracije pa smo zajeli še 9 dodatnih slik. Za njeno pravilno delovanje morajo namreč biti slike zajete pod čimbolj različnimi koti. Uporabljene fotografije so prikazane na sliki 5.1.



Slika 5.1: (a) Slike uporabljene pri kalibraciji in redki rekonstrukciji. (b) Dodatne slike uporabljene samo pri kalibraciji.

Rekonstrukcija je pridobljena do velikosti natančno. Posledično ima vektor med prvim parom kamer dolžino ena, razlog za to pa smo opisali v poglavju 3.4.1. Velikosti poravnamo tako, da translacijske vektorje med kamerami pomnožimo z razdaljo med prvim parom referenčnih kamer. Poravnati moramo še koordinatna sistema pridobljenih in referenčnih kamer. Ustrezno transformacijo pridobimo tako, da poračunamo rotacijsko matriko in translacijski vektor med prvo kamero rekonstrukcije in prvo referenčno kamero. Poravnani sistem je prikazan na sliki 5.2.

Napako rotacije pridobimo tako, da iz rotacijske matrike poračunamo rotacije okoli vsake izmed treh osi. Vrednosti absolutnih razlik nato povprečimo. Napaka translacije je razdalja med središčema kamer. Dobljeni

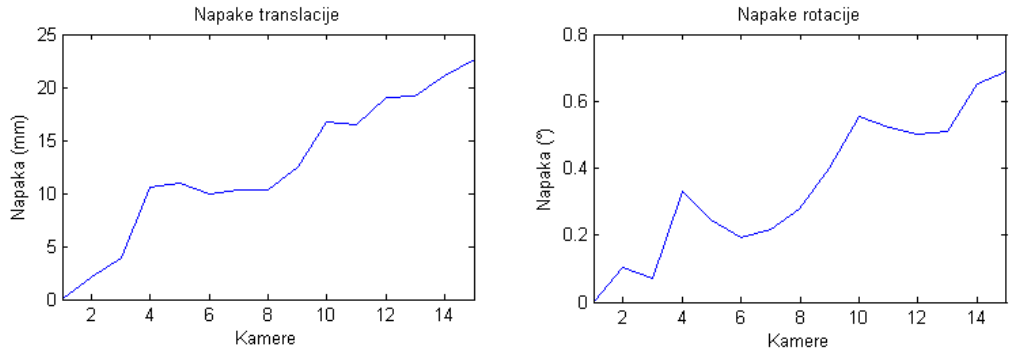


Slika 5.2: Z zeleno so obarvane referenčne, z rdečo pa rekonstruirane kamere. V sredini so prikazane tudi dodatne kamere, ki so bile uporabljene samo za kalibracijo. Opazimo lahko, da je prvih osem kamer rekonstrukcije dobro poravnanih, pri preostalih pa so odstopanja bolj opazna.

rezultati so povzeti v tabeli 5.1 in prikazani v obliki grafa na sliki 5.3. Povzamemo lahko, da pri računanju zunanjih parametrov kamer ne pride do prevelikih napak. Povprečna napaka translacije znaša 12.39mm, povprečna napaka rotacije pa samo 0.35°. Iz rezultatov lahko razberemo, da napaka narašča linearno. To je posledica akumulacije napake pri dodajanju pogledov. Kljub temu so lege kamer pridobljene dovolj natančno za nadaljnjo rekonstrukcijo.

Kamera	Napaka translacije [mm]	Napaka rotacije [°]
1	0.00	0.00
2	2.13	0.10
3	3.97	0.07
4	10.57	0.33
5	10.92	0.25
6	9.99	0.19
7	10.37	0.22
8	10.26	0.28
9	12.49	0.40
10	16.71	0.55
11	16.46	0.52
12	19.04	0.50
13	19.21	0.51
14	21.09	0.65
15	22.58	0.69
Povprečje	12.39	0.35

Tabela 5.1: Napaka lege rekonstruiranih kamer v primerjavi z referenčnimi legami.



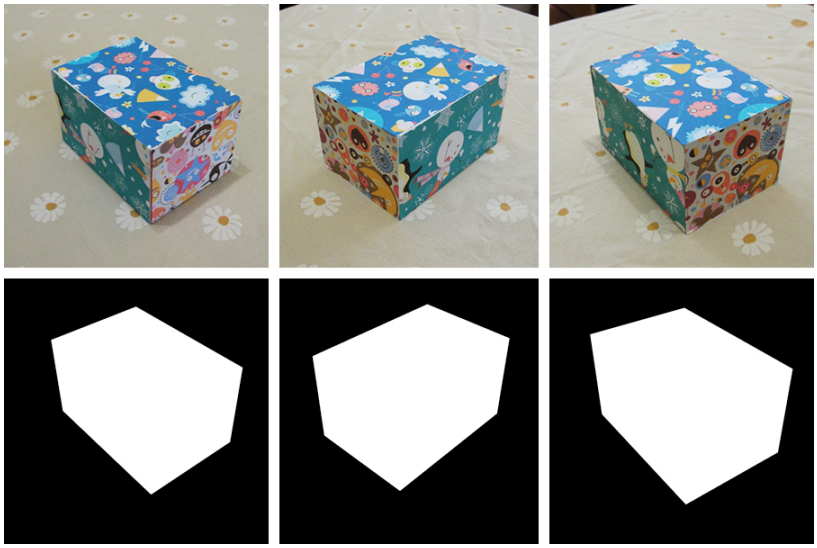
Slika 5.3: Grafični prikaz napak. Opazimo, da z dodajanjem kamer pride do akumulacije napake.

5.2 Kvantitativna evalvacija rekonstrukcije

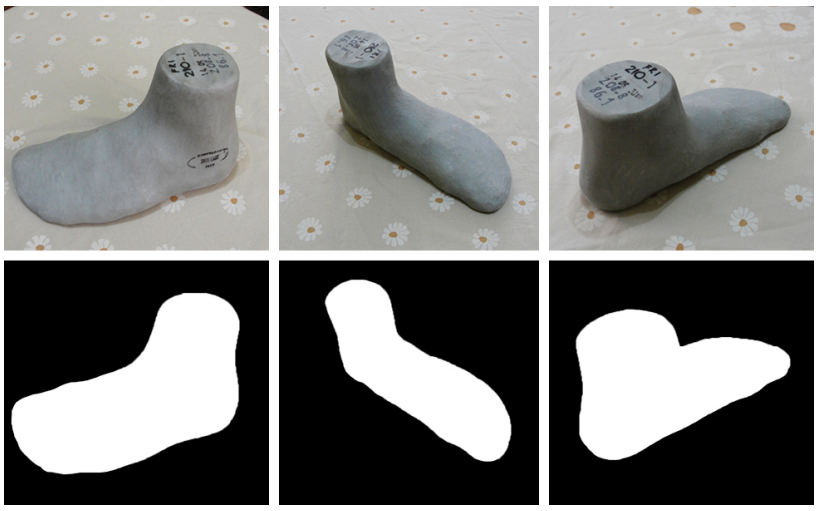
Pri kvantitativni evalvaciji je bila uporabljena enaka metoda kot jo je v svoji magistrski nalogi uporabil Gec [12]. Pri evalvaciji je bil uporabljen tudi del izvorne kode omenjenega avtorja. Natančnost goste rekonstrukcije smo evalvirali tako, da smo pridobljene modele primerjali z referenčnimi. 3D model je v našem primeru predstavljen kot gost oblak točk. Z referenčnim smo ga primerjali tako, da smo ju najprej poravnali, nato pa za vsako točko izračunali napako. Na voljo smo imeli referenčna modela kvadra in stopala, ki smo ju rekonstruirali tudi z našim postopkom. Dimenzije prvega so 70mm $\times$ 100mm $\times$ 60mm, drugega pa 209mm $\times$ 87mm.

Pri rekonstrukciji smo želeli pridobiti samo točke, ki so del objekta, zato smo ozadje odstranili s pomočjo maske. Le-ta objekt loči od ozadja, določili pa smo jo ročno. Dobljeni model je potrebno še poravnati z referenčnim. V ta namen uporabimo algoritem ICP (angl. Iterative closest point). Omenjeni algoritem iterativno poišče togo transformacijo, ki poravna podana modela. Deluje tako, da poizkuša zmanjšati razdaljo med točkami, podrobneje pa je opisan v [5]. Naši modeli so pridobljeni do velikosti natančno, zato moramo rešiti še eno težavo. Algoritem ICP namreč poišče le rotacijo in translacijo med podanima modeloma. Velikost v našem primeru določimo ročno tako, da jo vizualno čim bolje ocenimo na podlagi referenčnega modela. Poravnavo izboljšamo z ICP in po potrebi še enkrat popravimo velikost. Ker so naši modeli poravnani z referenčnimi, lahko napake izražamo v milimetrih.

Za redko rekonstrukcijo kvadra smo uporabili 24 vhodnih slik. Nato smo za gosto rekonstrukcijo izbrali tri pare slik in kalibriranih kamer, s katerimi smo pridobili delne rekonstrukcije in jih združili v celoten model. Določili smo tudi maske, ki objekt ločijo od ozadja. Skupaj z vhodnimi slikami so prikazane na sliki 5.4. Pridobljena rekonstrukcija in referenčni model sta prikazana na sliki 5.6. Pri rekonstrukciji stopala smo uporabili 27 vhodnih slik. Za gosto rekonstrukcijo pa smo ponovno izbrali tri pare slik, ki so prikazane na sliki 5.5. Dobljena rekonstrukcija in referenčni model sta prikazana na sliki 5.7.



Slika 5.4: Zgornji del prikazuje slike uporabljene pri gosti rekonstrukciji kvadra. Prikazane so samo leve slike parov. Spodnji del prikazuje maske uporabljene za odstranitev ozadja.



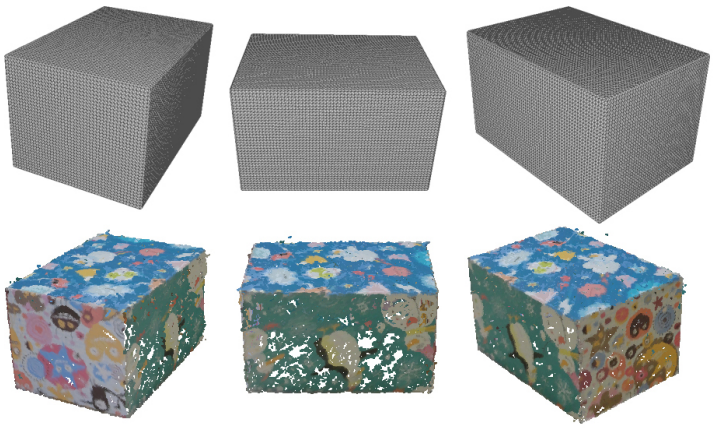
Slika 5.5: Zgornji del prikazuje slike uporabljene pri gosti rekonstrukciji stopala. Prikazane so samo leve slike parov. Spodnji del prikazuje maske uporabljene za odstranitev ozadja.

Dobljeni rekonstrukciji imata preveč točk, da bi ju lahko v celoti uporabili pri evalvaciji. Velikost modelov je namreč problematična za branje in prikazovanje v Matlab-u. Za nadaljnje računanje modela poenostavimo, in sicer tako da v obeh primerih ohranimo 40.000 naključnih točk. Z opisanim postopkom poravnamo rekonstruirana modela z referenčnimi. Rezultat poravnave za kvader je prikazan na sliki 5.8, za stopalo pa na sliki 5.9. Zaradi prekrivanja modrih in rdečih točk lahko ocenimo, da sta modela dobro poravnana, kljub temu, da smo velikosti nastavili ročno.

Porazdelitev napak za model kvadra je prikazana na sliki 5.10. Rekonstrukcija tega modela je bila precej uspešna. Opazimo lahko, da je zelo malo točk, ki odstopajo za več kot 3mm. To potrjuje tudi histogram napak na sliki 5.12. Poleg tega so redke tudi osamele točke. Povprečna napaka tega modela znaša 0.99mm. Rekonstrukcija stopala ima več napak v primerjavi s kvadrom. Porazdelitev napake je prikazana na sliki 5.11. Pojavi se tudi nekaj osamelih točk, kar je razvidno iz histograma napak, na sliki 5.13. Slabši rezultati so bili pričakovani, saj je model bolj kompleksen in njegova tekstura ni preveč izrazita, to pa otežuje iskanje ujemanj. Povprečna napaka znaša 2.06mm, ostali podatki o napaki za oba modela, pa so povzeti v tabeli 5.2.

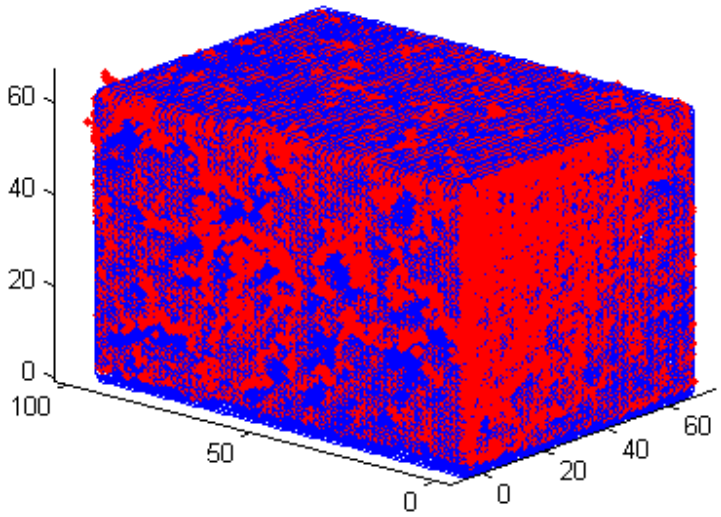
Predmet	Povprečna napaka [mm]	Mediana [mm]	Standardni odklon [mm]	Največja napaka [mm]
Kvader	0.99	0.92	0.47	33.57
Noga	2.06	1.85	1.17	10.49

Tabela 5.2: Povprečna napaka obeh modelov.

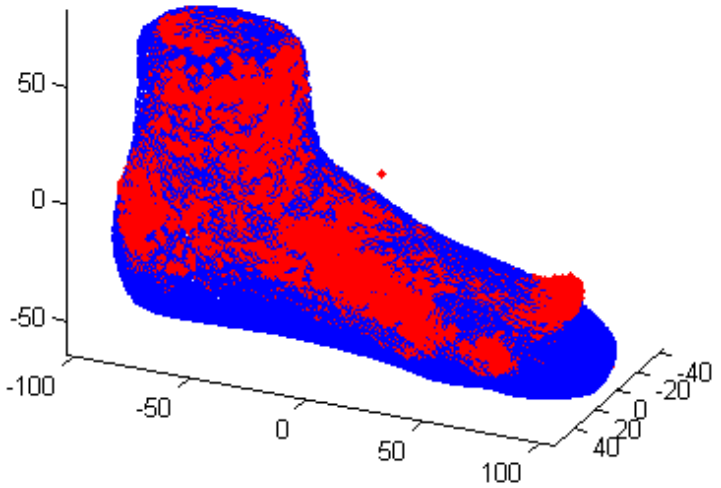


Slika 5.6: Zgoraj je prikazan referenčni model kvadra (13.500 točk), spodaj pa pridobljena rekonstrukcija (236.373 točk).

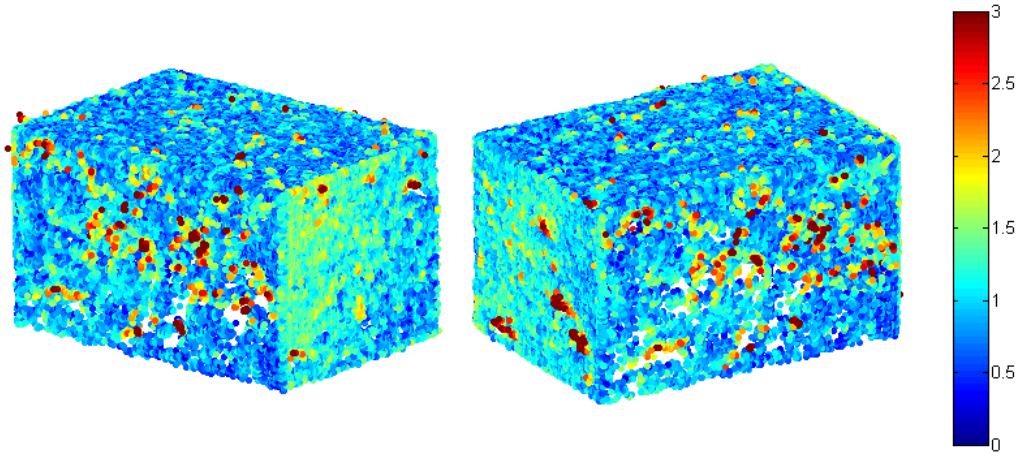
Slika 5.7: Zgoraj je prikazan referenčni model stopala (5.977 točk), spodaj pa pridobljena rekonstrukcija (226.259 točk).



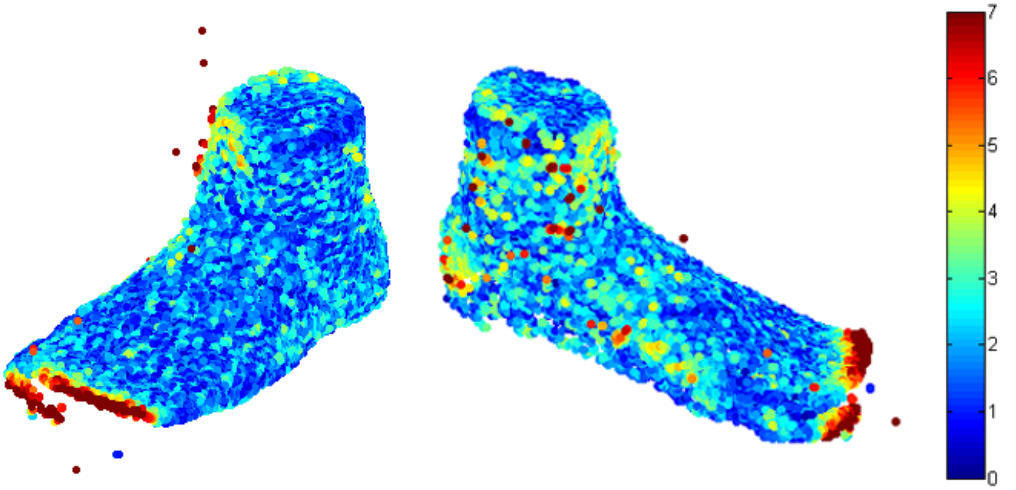
Slika 5.8: Poravnava modela kvadra (rdeča) z referenčnim modelom (modra).



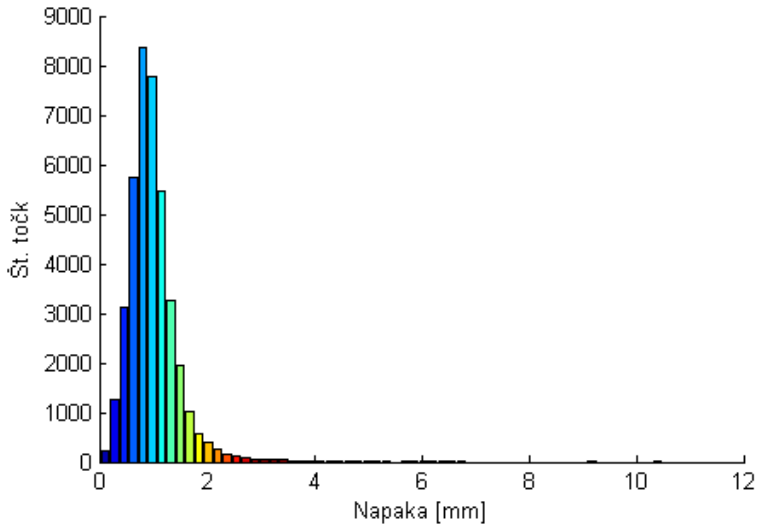
Slika 5.9: Poravnava modela stopala (rdeča) z referenčnim modelom (modra).



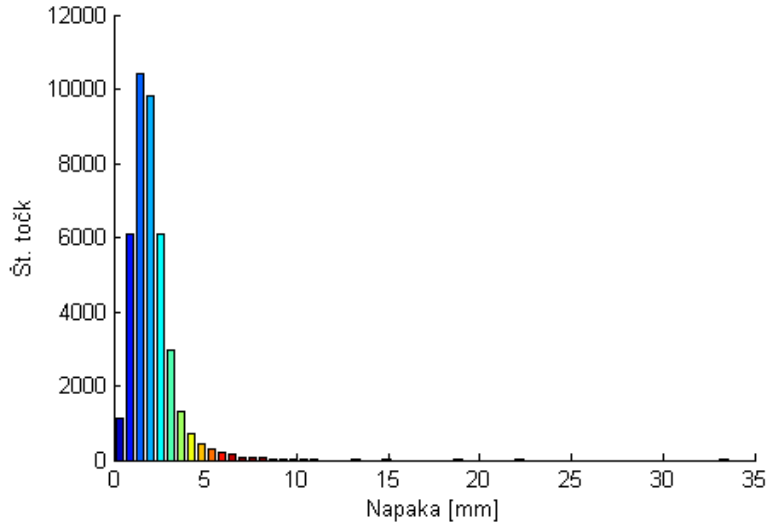
Slika 5.10: Porazdelitev napak na modelu kvadra.



Slika 5.11: Porazdelitev napak na modelu stopala.



Slika 5.12: Histogram napak modela kvadra.



Slika 5.13: Histogram napak modela stopala.

5.3 Primeri rekonstrukcij

Podali bomo še nekaj primerov rekonstrukcij. Izbrali smo si pet predmetov in zgradili njihove 3D modele. Predmeti so bili segmentirani iz ozadja tako, da smo slikam ročno določili masko, ki določa katere točke na sliki želimo rekonstruirati. Izbrani predmeti so bili čevelj, stol, kartonska embalaža, predalink in kalkulator. V enakem zaporedju so prikazani na slikah 5.14, 5.15, 5.16, 5.17 in 5.18. Pri rekonstrukciji dobro teksturiranih predmetov (čevelj, stol, kartonska embalaža) naš postopek nima večjih težav. Večje nepravilnosti se pojavijo zaradi odbojev svetlobe na reflektivnih površinah. To je vidno na modelu predalnika. Njegova rekonstrukcija ima zaradi lakiranega lesa dosti osamelih točk. Enak pojav opazimo tudi na ekranu kalkulatorja. Na modelih lahko opazimo tudi, da imajo nekateri neenakomerno porazdelitev točk in nekaj lukenj. To bi lahko do neke mere odpravili z bolj pozornim zajemanjem slik, prava rešitev pa bi bila izboljšanje postopka za dodajanje kamer v sistem.

Slika 5.14: Model čevlja. Za gosto rekonstrukcijo so bili uporabljeni trije pari kamer, končni model pa ima 231.863 točk.

Slika 5.15: Model stola. Za gosto rekonstrukcijo sta bila uporabljena dva para kamer, končni model pa ima 227.171 točk.

Slika 5.16: Model kartonske embalaže. Za gosto rekonstrukcijo so bili uporabljeni trije pari kamer, končni model pa ima 277.845 točk.

Slika 5.17: Model predalnika. Za gosto rekonstrukcijo sta bila uporabljena dva para kamer, končni model pa ima 223.907 točk.

Slika 5.18: Model kalkulatorja. Za gosto rekonstrukcijo sta bila uporabljena dva para kamer, končni model pa ima 105.420 točk.

Poglavje 6

Sklepne ugotovitve

V tem diplomskem delu smo predstavili postopek za rekonstrukcijo objektov na podlagi zajetih slik. Najprej smo opisali teoretično ozadje, ki je skupno takšnim oblikam rekonstrukcije. Nadaljevali smo z opisom algoritmov, ki so bili uporabljeni kot gradniki pri pridobivanju 3D modelov. Naš postopek smo razdelili na tri večje dele. S kalibracijo kamer smo pridobili njihove notranje parametre. Gradnjo 3D modela smo pričeli z redko rekonstrukcijo. Realizirali smo jo tako, da smo najprej na slikah poiskali značilnice, nato pa s pomočjo notranjih parametrov poračunali lege kamer in lokacije značilnic v prostoru. V zadnjem koraku goste rekonstrukcije smo uporabili lege kamer, zato da smo z njimi poenostavili iskanje ujemanj med slikami in tako poračunali globinske slike. Iz njih smo pridobili delne modele in jih združili v končni 3D model.

Evalvacija rezultatov je pokazala, da z našim programom pridobimo pričakovane rezultate. Zanimala nas je natančnost poračunanih zunanjih parametrov, ter natančnost rekonstrukcije. Pri evalvaciji zunanjih parametrov kamer smo ugotovili, da pri zaporednem dodajanju kamer pride do akumulacije napake. Lege kamer so kljub temu določene dovolj natančno, da so primerne za gosto rekonstrukcijo. Natančnost rekonstrukcije smo preverili tako, da smo pridobljene modele, primerjali z referenčnimi modeli. Na voljo sta nam bila dva referenčna modela, in sicer kvader in stopalo. Povprečna napaka je tu znašala le nekaj milimetrov, kar pomeni, da so pridobljeni modeli relativno

natančni. Na koncu smo podali še nekaj primerov rekonstrukcij.

Naša implementacija ima še nekaj prostora za izboljšave. Zelo koristna izboljšava bi bila zmanjšanje potrebe po uporabniški interakciji. Notranje parametre kamere je mogoče pridobiti tudi direktno iz slik s postopkom samodejne kalibracije [22]. Z uporabo takšnega postopka, ročna kalibracija ne bi bila več potrebna. Uporabnik mora posredovati tudi pri gosti rekonstrukciji tako, da izbere pare kamer, ki se mu zdijo najbolj primerni za grajenje globinske slike, in nastaviti ustrezne parametre . Ta del bi verjetno lahko izboljšali tako, da bi se dobri pari detektirali avtomatsko, parametri pa bi se poračunali na podlagi lege kamer.

Dodajanje nove kamere v sistem bi lahko izboljšali tako, da bi jo dodajali na podlagi vseh bližnjih kamer. Tako kamere ne bi bilo potrebno premikati samo v eno smer. Akumulacijo napake bi lahko preprečili z metodo zapiranja zank (angl. loop closing). Ta deluje tako, da ob detekciji že znanega pogleda popravi vse vmesne lege kamer. Postopek je uporabljen tudi na področju robotike, in sicer v algoritmu SLAM [9] (Simultaneous localization and mapping), ki se uporablja za lokalizacijo robota. Natančnost modelov bi lahko izboljšali tudi z uporabo nekaterih bolj naprednih algoritmov, ki za grajenje globinske slike uporabljajo več kamer.

Za naš sistem lahko povzamemo, da 3D modele pridobi z dobro natančnostjo, predstavlja pa tudi dober gradnik za nadaljnji razvoj in izboljšave.

Literatura

[1] Knjižnica mexopencv. `http://kyamagu.github.io/mexopencv`. Pridobljeno: 30-08-2015.

[2] Program MeshLab. `http://meshlab.sourceforge.net/`. Pridobljeno: 08-09-2015.

[3] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 510–517. Ieee, 2012.

[4] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[5] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Robotics-DL tentative*, pages 586–606. International Society for Optics and Photonics, 1992.

[6] Jean-Yves Bouguet. Orodje Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib_doc/index.html`. Pridobljeno: 06-09-2015.

[7] Jean-Yves Bouguet and Pietro Perona. 3D photography on your desk. In *Computer Vision, 1998. Sixth International Conference on*, pages 43–50. IEEE, 1998.

[8] Gary Bradski and Adrian Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008.

[9] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

[10] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[11] Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):930–943, 2003.

[12] Sandi Gec. Razvoj sistema za 3D rekonstrukcijo predmetov z uporabo pasivnih metod računalniškega vida. Master's thesis, Fakulteta za računalništvo in informatiko, 2015.

[13] Pascal Getreuer. Knjižnica PLY_IO. `http://people.sc.fsu.edu/~jburkardt/m_src/plyio/ply_io.html`.
Pridobljeno: 08-09-2015.

[14] Richard Hartley et al. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, 1997.

[15] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[16] Richard I Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[17] Kiriakos N Kutulakos and Steven M Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[18] Taehee Lee. Knjižnica VLG. `http://vision.ucla.edu/vlg`.
Pridobljeno: 30-08-2015.

[19] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2548–2555. IEEE, 2011.

[20] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[21] Marc Pollefeys. Visual 3D modeling from images. In *VMV*, page 3, 2004.

[22] Marc Pollefeys and Luc Van Gool. Stratified self-calibration with the modulus constraint. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):707–724, 1999.

[23] Marc Pollefeys, Luc Van Gool, Maarten Vergauwen, Frank Verbiest, Kurt Cornelis, Jan Tops, and Reinhard Koch. Visual modeling with a hand-held camera. *International Journal of Computer Vision*, 59(3):207–232, 2004.

[24] Carlo Tomasi and Takeo Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[25] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision algorithms: theory and practice*, pages 298–372. Springer, 2000.

[26] A. Vedaldi and B. Fulkerson. Knjižnica VLFeat. `http://www.vlfeat.org`.
Pridobljeno: 30-08-2015.

[27] A. Vedaldi and B. Fulkerson. Vadnica za VLFeat. `http://www.vlfeat.org/overview/sift.html`.
Pridobljeno: 02-09-2015.

[28] Li Zhang, Brian Curless, and Steven M Seitz. Rapid shape acquisition using color structured light and multi-pass dynamic programming. In *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pages 24–36. IEEE, 2002.

[29] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. *International journal of computer vision*, 27(2):161–195, 1998.

[30] Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

[31] Zhengyou Zhang. Microsoft Kinect sensor and its effect. *MultiMedia, IEEE*, 19(2):4–10, 2012.